PMCprimo C

Motion Control Systems

Programming Manual - 1002926-EN-02

Dieses Dokument ist das Originaldokument.

Alle Rechte an dieser Dokumentation sind der Pilz GmbH & Co. KG vorbehalten. Kopien für den innerbetrieblichen Bedarf des Benutzers dürfen angefertigt werden. Hinweise und Anregungen zur Verbesserung dieser Dokumentation nehmen wir gerne entgegen.

Pilz®, PIT®, PMI®, PNOZ®, Primo®, PSEN®, PSS®, PVIS®, SafetyBUS p®, SafetyEYE®, SafetyNET p®, the spirit of safety® sind in einigen Ländern amtlich registrierte und geschützte Marken der Pilz GmbH & Co. KG.

SD bedeutet Secure Digital

1 General 1-6
1.1 Copyright 1-6
1.2 Note: 1-6
1.3 Previous editions: 1-6
1.4 Codes and symbols 1-7
1.5 Preface 1-8
1.6 General description 1-9

2 Command overview 2-11
2.1 General information about the instruction set 2-11
2.2 Command execution 2-15

3 Command description 3-17
3.1 Data commands 3-17
3.2 Mode commands 3-32
3.3 Motion commands 3-33
3.4 Parameter commands 3-44
3.5 Program commands 3-58
3.6 Mapping commands 3-73
3.7 Wait loops 3-99
3.8 Monitoring functions 3-106
3.9 Control parameters 3-112
3.10 Reference commands 3-116
3.11 Digital input and output commands 3-137
3.12 Analogue input and output commands 3-155
3.13 Definition commands 3-161
3.14 Display commands 3-187
3.15 Phase commands 3-199
3.16 Web tension control with analogue input 3-203
3.17 Variable commands 3-211
3.18 Commands for CANopen network 3-224
3.19 Commands for the safety card 3-227

4 Status and error messages 4-229
4.1 Status messages 4-229
4.2 Error messages 4-230
4.3 Coded status messages 4-253

5 Interface: PMCprimo <-> Machine 5-254
5.1 Installation guidelines 5-254
5.2 Monitoring functions 5-255
5.3 Encoders 5-255
5.4 Setpoint output 5-255
5.5 Analogue auxiliary output 5-255
5.6 Ethernet interface 5-256
5.7 Inputs and outputs 5-256

5.8 Drive commands 5-262
5.8.1 CAN networking 5-263

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Figure 1: Trapezoidal velocity 3-34
Figure 2: Course of position with trapezoidal velocity profile 3-34
Figure 3: Triangular velocity profile 3-35
Figure 4: Movement ended via the ST command 3-37
Figure 5: Movement ended via the AB command 3-38
Figure 6: Moving at constant velocity 3-39
Figure 7: Initialisation with return to reference 3-41
Figure 8: Initialisation of position and cycle length 3-43
Figure 9: Changing travel velocity while moving 3-44
Figure 10: High and low acceleration value 3-45
Figure 11: Set braking ramp 3-46
Figure 12: Positioning with creep speed 3-48
Figure 13: Velocity levels SV/SS 3-49
Figure 14: Monitoring time “TI” 3-51
Figure 15: Simple position mappings 3-74
Figure 16: Position mapping for a defined area 3-75
Figure 17: Position mapping for a cyclical machine 3-76
Figure 18: Effects of position offset 3-77
Figure 19: Example of software differential 3-78
Figure 20: Example of software differential 3-78
Figure 21: Position mapping as equation 3-79
Figure 22: Position mapping with differential as equation 3-79
Figure 23: Software clutch behaviour 3-96
Figure 24: Time delay 3-99
Figure 25: Wait for input line in group g 3-100
Figure 26: Wait for absolute position 3-101
Figure 27: Wait for relative position 3-102
Figure 28: Relationship between lag distance and setpoint output 3-113
Figure 29: Interaction between the commands when a reference signal is detected 3-117
Figure 30: Interaction between the reference error correction commands 3-118
Figure 31: Cycle length 3-121
Figure 32: Reference error correction with RV as velocity gain 3-126
Figure 33: Reference error correction with RV as path 3-127
Figure 34: Variable distance for reference error correction 3-130
Figure 35: Special filter function for referencing 3-131
Figure 36: Limit values for evaluating the reference input 3-132
Figure 37: Monitor function on the analogue auxiliary output 3-158
Figure 38: Example 1 for PO command 3-173
Figure 39: Example 2 for PO command 3-174
Figure 40: Velocity-dependent phase displacement 3-200

1 General

1.1 Copyright

Copyright 2009 Pilz GmbH & Co. All rights reserved.

All rights reserved. No part of this document may be reproduced in any form (by printing, photocopying, microfilm or any other method), nor may it be processed, duplicated or distributed by electronic means without written permission from Pilz GmbH & Co. KG.

1.2 Note:

Pilz GmbH & Co. KG reserves the right to amend this document at any time.

The given examples are for illustration only. We accept no responsibility as to their suitability for specific applications. This manual has been produced with great care, but we cannot accept liability for errors or omissions. We welcome your suggestions as to how to improve either our products or our documentation.

1.3 Previous editions:

Edition	Comment
PH.V1.0.D	For PMCprimo versions up to and including 3.10

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

1.4 Codes and symbols

Code	Meaning/description
PMCprimo	Registered trademark of Pilz GmbH & Co. KG
PMCtendo	Registered trademark of Pilz GmbH & Co. KG

Symbol	Meaning/description
	This symbol signifies a potential threat to life and health. Failure to comply with this information can cause harmful health consequences, including life threatening injuries.
	This symbol indicates an example which is intended to illustrate the technical point.
	Under this symbol you'll find applications tips and particularly useful information. This helps you to use all the machine functions at optimum level.
•	Listing

1.5 Preface

This manual describes the complete function range available for programming PMCprimo motion control systems.

PMCprimo systems are equipped with an Ethernet interface for programming. This interface is used to connect PMCprimo to a PC. Communication with the control system is via a terminal program. Easy-to-learn commands enable the user to program and commission PMCprimo quickly and effectively.

As an alternative to the programming language described here, PMCprimo motion control systems can also be fully programmed under an IEC 61131-3 interface (CoDeSys). There is a separate manual for this, describing the functions and commands of the programming language (user manual for PMCprimo SoftPLC).

Please read this manual carefully.
This manual is the ***Reference Manual*** for all PMCprimo user commands, even if these are not described in other documents, or are even contradicted in other documents!

When used correctly, PMCprimo motion control systems use electronic axis synchronisation to resolve tasks that have previously been implemented primarily by mechanical components. The process uses electronic cam disks, cam mechanisms and other functions to enable the controlled, highly dynamic operation of servo drives.

PMCprimo motion control systems also offer the option to assume complete machine control via soft PLC functionality.

You will need an understanding of the system's basic principles in order to do this. This manual aims to provide this understanding. Another effective way is to attend a Pilz training course.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

1.6 General description

This chapter contains a brief description of the possibilities of PMCprimo motion control systems.

PMCprimo motion control systems are available in various hardware versions and can control up to 60 axes.

PMCprimo is programmed in a high level language or under IEC 61131-3 (CoDeSys). The central communication medium is a terminal program (see user manual "PMCprimo Motion Control Tools") on a PC, which is connected to PMCprimo via an Ethernet interface. All parameter settings and user programs can be stored in PMCprimo in a Flash memory or on a memory card in PMCprotego D.

Software functions for moving servo axes:

The PMCprimo commands described here are used to set the parameters in the control system and to program functions for the respective application.

A large selection of commands enables the implementation of complex applications. PMCprimo has commands that span more than one axis. In other words, a single command can affect multiple axes simultaneously.

Reference commands enable automatic initialisation and reference error correction as the machine is running. The reference error can be compensated in a variety of ways, e.g. through definable ramp or time functions.

As a result, any deviations in the product or machine parameters can be corrected flexibly (based on the cycle).

Freely definable Master-Slave relationships enable the user to define any velocity profiles for motors. The use of shock-free motion sequences (e.g. modified sinoide) keeps the mechanical load on the machines as low as possible.

PMCprimo can equip motors with an electronic gear function through software technology. In addition to the gear function, the relationship between the motor positions can be freely defined by the user (linear and non-linear correlations via tabular position mapping). A large number of VDI-defined motion types are available for selection. Tabular position mappings can replace mechanical cam disks and/or cam mechanisms.
If necessary, PMCprimo can use the internal motion generator to create new position mappings as the machine is running. As such, product-based machine conversions are realised at the touch of a button.
In addition to a wide range of options for synchronising machine axes, the user also has access to a series of commands for absolute, relative or infinite positioning of drives.

All axes can also be operated in a virtual mode (electronic main shaft) for commissioning or as auxiliary axes.

1.6 General description

Hardware functions:

PMCprimo is designed for the use of incremental or absolute encoder systems with digital step values. Incremental rotary encoder signals can be evaluated fourfold. Alternatively, rotary encoders with SSI or CAN-open interface can also be used, as absolute or incremental encoders.

All rotary encoder inputs are always isolated via optocouplers. They are designed for rotary encoders that have potentially isolated line driver outputs, to guarantee the safe transfer of information in an industrial environment.

PMCprimo has freely programmable digital inputs and outputs (number depends on the model). All digital inputs can trigger an alarm in the event of a change of state. The reaction in the event of an alarm is defined by the user. Inputs can be defined to start a program in the event of a change of state. Digital outputs can be defined for various purposes, such as electronic cams, error outputs, reference error output, etc.

One of the functions of the inputs and outputs is to serve as an interface to a PLC. They are potential-free and operate with 24 VDC. MODBUS TCP/IP or diverse fieldbus systems are available for easy communication with a PLC or any other master computer.

Applications:

Typical application fields for PMCprimo motion control systems are:

- Packaging plant and machinery
- Printing and paper machinery
- Handling systems
- Production machinery

etc.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

2 Command overview

2.1 General information about the instruction set

On PMCprimo, a command name always consists of two letters. A command may also have a varying number of parameters. These are stated with "n", "nn" or "m" in this manual. Binary parameters (0 and 1) are identified with "bb". All input commands and command sequences must be acknowledged with a RETURN "<CR>".

PMCprimo sends back a prompt after each RETURN. The prompt is composed as follows:

```
0.1:
```

0	**.**	**1**	**:**
Current network subscriber. This is always '0'.	Separator	Current axis	Current state of axis (see Status messages, page 4-229).

If an axis has not previously been stated, a command will always refer to the current axis displayed in the prompt.

To query the value of a command, the command must be entered without parameters.

Example 1 (enter value):

```
SV2000 <CR>
```

Example 2 (query value):

```
SV<CR>
SV=2000
```

Numeric parameters can be input and output either in decimal or hexadecimal format. A hexadecimal entry must be preceded by “0x”. Negative hexadecimal figures are represented with a minus symbol and not as a two's complement.

Leading zeros can be omitted when entering “Bit values”.

2.1 General information about the instruction set

Names for variables, programs and mappings consist of the letters "A-Z", the numbers "0-9" and the character "_". However, they may not start with a number. Entries are not case-sensitive. The system automatically converts lower case letters into upper case.

A command line may be made up of several commands.

Example:

```
0.1>ch2;mr2000;wt500<CR>
```

The commands are separated by the character "**;**". The maximum line length is 200 characters. The Escape key can be used to cancel an output command or the "??" command for help ("??" command see page 3-197).

Any '+' sign at the start of an input line is ignored. This is necessary because a modem connection is ended by the ASCII sequence "+++", whereby one '+' character remains in the input buffer.

PMCprimo has an extensive instruction set. The commands can be grouped into the following categories:

- Data commands (Chapter 3.1)
  Axis selection and handling stored data.
- Mode commands (Chapter 3.2)
  Activate / deactivate position control loop
- Motion commands (Chapter 3.3)
  Basic commands for motor start and stop, using a trapezoidal or sinusoidal velocity profile.
- Parameter commands (Chapter 0)
  Input and output of axis parameters, such as velocity, acceleration, etc.
- Program commands (Chapter 3.5)
  Commands that are important for the input and output of programs and how they are created.
- Mapping commands (Chapter 3.6)
  Creation and execution of tabular position mappings between individual axes for Master-Slave mode.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

2.1 General information about the instruction set

- Wait loops (Chapter 3.7)
  Programming of wait loops within a program or command sequence, to query time elapsed, reaching a position, state of an input, etc.

- Monitoring functions (Chapter 3.8)
  Setting of error limits and definition of the PMCprimo's reaction in the event of an error.

- Control parameters (Chapter 3.9)
  Setting of all factors that influence the control loop, such as KP, KI, KD, KV and KF.

- Reference commands (Chapter 3.10)
  Activation and setting of the reference behaviour.

- Digital input and output commands (Chapter 3.11)
  Querying and linking of digital inputs and setting / resetting of digital outputs. Using timers and counters in conjunction with digital inputs and outputs.

- Analogue input and output commands (Chapter 3.12)

- Definition commands (Chapter 3.13)
  Definition of digital input and output functions, e.g.: input is reference input or output is electronic cam.

- Display commands (Chapter 3.14)
  Display of data such as position, velocity, lag distance and time.

- Phase commands (Chapter 3.15)
  Velocity-dependent phase displacement with synchronous running and electronic cams.

- Additional control with analogue input (Chapter 3.16)
  These commands enable a motor to be operated with web tension control.

- Variable commands (Chapter 3.17)
  All numeric values can also be replaced by variables. Variables can be used to perform arithmetic operations. Specific variables (bus variables) can also be read via an operator terminal or bus system and can be modified in PMCprimo.

- CAN-open commands (3.18)

- Commands for the safety card (3.19)

2.1 General information about the instruction set

In the PMCprimo it is possible to store variable values in an MRAM. This has the advantage that the current variable values are still available after switching off and on. The current rotary encoder value can also be stored in RAM.

The PMCprimo can access an SD card on a PMCprotego, on which all the data can also be stored.

Drive parameters for the PMCtendo or PMCprotego can be entered directly via the software PDrive or Pterm.

All positional entries are made in rotary encoder pulses (increments)

Data and parameter values are stored in a Flash memory with the “SP” command. More than 500 MB are available for storing programs. Variables are also stored dynamically in this memory area.

The stored data is checked via check sum as the control system is started up. If a fault is detected, an error message appears and the stored data is rejected. The control system will then load the default setting. A fault may occur if PMCprimo is switched off during the save process, for example.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

2.2 Command execution

Commands can be executed in various ways in PMCprimo. The methods for command execution are:

Command line

Commands can be entered either individually or as a command line via the Telnet interface (Ethernet); the command is then executed using <CR> (Enter key). Individual commands are executed immediately. If command lines have been entered, the individual commands are processed in order. Several command lines can be processed in parallel.

Example for individual commands:

`0.1>MA1000<CR>`	PMCprimo moves the motor to the absolute position of 1000 increments

Example for command line:

`0.1>MA1000;WT50;MA0<CR>`	PMCprimo moves the motor to the absolute position of 1000 increments, waits 50 x 1 ms and then moves the motor to the absolute position 0.

Programs

A program is a combination of multiple commands. The commands in a program are processed consecutively, as on a command line. A program is defined via the ES command, whereby NS completes the entry. A program is called via the "**XS**" command. Programs can then call other programs. Programs can also run in parallel via the "**XP**" command or by starting via trigger variables or trigger inputs. A program that is already active cannot be started a second time.

2.2 Command execution

Example:

```
0.1> ES PROG1<CR>                          Definition of program PROG1
002E CH1;MA1500;VC+;WT100;ST<CR>
003E NS <CR>                               End of PROG1
0.1>
```

```
0.1> ES PROG2<CR>                          Definition of program PROG2
002E CH1;DP;DV<CR>
003E NS<CR>                                End of PROG2
0.1>

0.1> XS PROG1<CR>                          Program call of PROG1
0.1> XS PROG2<CR>                          Program call of PROG2

0.1> DP=345                                Output of actual position of axis 1
0.1> DV=0                                  Output of instantaneous velocity
```

Input functions

Programs can be called via digital inputs. Mapping is carried out via the DI command.

Trigger variable

Commands or even programs can be called by activating a variable. The value of the variable must not change in the process.

Where commands cannot be executed by PMCprimo, e.g. a travel command when the position control loop is open, or in the event of a command collision, PMCprimo signals this to the user with an appropriate error.

Example for trigger variable:

```
0.1>$TRIGVAR>TEST<CR>      The variable TRIGVAR is defined as a trigger variable.
0.1>EV $TRIGVAR            The trigger variable is enabled.
0.1>$TRIGVAR=0             The program TEST is started by assigning a value to
                           the trigger variable.
```


Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3 Command description

3.1 Data commands

Commands used for data management and system configuration of PMCprimo.

CD Change operating mode

(**C**hange Operating Mo**d**e)

This command can be used to change the configuration of the PMCprimo control system. When CD is entered, the following menu appears:

```
0.1: cd
0.1:
A C T U A L   C O N F I G U R A T I O N:
*** Project Info PLC ********************
*** Date of Project:    2012-05-03
*** Project:            Neu_BasicProject_17_Juni_2010.pro
*** Title:              Basic project for test
*** Version:            V3.100
*** Author:             S. Stöckl
*** Description:
*****************************************
     Operate Mode:      STANDALONE
(24) Cycle Time:        1000 µs
( 4) Actual IP address: 192.168.0.11
( 4) Actual Netmask:    255.255.255.0
( 4) Actual Gateway:    192.168.0.1
(12) Number of Channels:30
( 8) CAN Cycle time:    4 ms
( 9) CAN node address   50
( 3) CAN1 baudrate:           1000 KBit
(21) CAN2 baudrate:           1000 KBit
(23) CAN2 cycle time:         4ms (equal CAN1)
(22) CAN-mode:                CAN1: Master  CAN2: Master
(26) PMCprotego with SD-Card: automatic
(11) PROFIBUS Address:        0
( 5) PROFIBUS IN/OUT Length:  0 (Words)
( 6) PROFIBUS Offset:         0
*****************************************
0: Exit menu
2: Delete application data
3: Change CAN1 baudrate
4: Change Ethernet
5: Change in/out length for Profibus
6: Change offset for Profibus
8: Set CAN Cycle time
9: Set CAN address
11: Change Profibus address
12: Change number of channels
21: Change CAN2 baudrate
22: Enable slave mode for CAN1/2
23: Set CAN2 cycle time
24: Change cycle time of system
26: Set address for PMCprotego with SD-Card
Choice [Return; ESC exits menu]:
```

3.1 Data commands

Explanation of the configuration options:

0: Exit the configuration menu by entering 0 or Esc.

2: This is where the program stored on the PMCprimo can be deleted.

3: Setting the baud rate for CAN1. The following baud rates are possible: 20/50/100/125/250/500/800/1000KBit

4: The IP address, gateway address and net masks can be set.

5: Setting the number of input/output words for Profibus.

6: An offset can be stated for the fieldbus address. The value range is between 0 and 100. The default value is 0. As a result it is possible to address the input banks 0.101 to 0.108 via the bus variable $B101 to $B108.

8: It is possible to modify the cycle time of the CAN bus. Possible settings are 1, 2, 4 (basic setting) and 8 milliseconds. In general, reducing the cycle time improves the motor's control behaviour.

9: It is possible to set the node address on the CAN bus. As a result, a CAN operator terminal can address the PMCprimo control system, for example.

11: Setting the Profibus address.

12: Setting the number of axes:
30 axes are defined by default. This number can be increased to a maximum of 60. However, in this case the processor should be clocked at sufficient speed.

21: Setting the baud rate for CAN2. If the baud rate is to be the same as that of CAN1, 0 must be set. This is also the factory setting.

22: Slave mode can be set on CAN1 or CAN2. This means that no network scan is performed on the CAN Slave and there is a reaction to SDO access.

23: An extra cycle time can be set for CAN2. If 0 is set (factory setting), the cycle time of CAN1 is adopted.

24: The system cycle time can be set. Select between 250, 500 and 100 µs. This time determines the sampling time for generating set values and position control.

26: A specific address can be stated on the PMCprotego for saving to an SD card.
If the address 0 is stated, when the SP or RD command is issued, the system automatically searches for the first PMCprotego with SD card and accesses this.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

The command has been expanded by 2 optional parameters: **CD [n,x].** These parameters enable a setting to be made via a program, without a selection menu.

Permitted values for n:

-2	All operating parameters are set to the factory setting if x = 1 IP address: 192.168.0.11 Net mask: 255.255.255.0 Gateway: 192.168.0.1 The stored programs are deleted if x = 1
3	Set CAN baud rate
5	Number of fieldbus words
6	Profibus offset
8	CAN cycle time
9	CAN node number
11	Profibus address
12	Number of axes; the factory setting (32) is adopted for x=0.
22	CAN Slave setting
23	Cycle time for additional CAN
24	System cycle time
26	Address for SD card

3.1 Data commands

VNnn Print version number

(Print **v**ersion **n**umber)
Permitted values for nn: 0 to 1

n = 0: Lists the current software version, serial number, material number and product version number.

```
0.1: vn
0.1 PMCprimo C Channels 30, Version 3.0.2 May 10 2012 13:34:11, Mat.Nr:680055, Ser.Nr.:111,
Pr.Ver:1.0
```

n = 1: Also lists other version numbers, configuration and MAC addresses, in addition to the current software version, serial number, material number and product version number.

```
0.1> vn1
0.1>
Version:              3.1.0_Jan 10 2013 08:55:25
Material Number:      680050
Serial Number:        100004
Product Version:      0.1
HW-Version:           UNDEF
PriBoLo:              v02.01
FPGA Driver:          v02.12
FPGA-File CAN/CAN:    v02.01.00
BSP-Version:          v01.02
HW-Type:              PMCprimo C    600MHz    CAN/CAN
HW-Revision:          1
CPU-Family:           0x006    Model:0x26    SteppingID:0x1
BIOS creation date:   08/16/2011
MRAMData:             Ver:004    Size:524288    Reserved:160300
MAC primo:            00:02:48:40:C9:2D
MAC COM-Modul:        00:02:48:40:C7:A9
```


Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

HWnn Print hardware

(Print **h**ard**w**are)

Permitted values for nn: 0 to 2

n = 0: All CAN devices are listed.

n = 1: Hardware information is listed as follows:

```
SOFTWARE
    Firmware:       3.0.2_T05, May 10 2012, 12:56:18
    Motion:         INSTALLED
    IEC PLC:        INSTALLED
    Interpolation:  NOT AVAILABLE

ETHERNET
    IP address      192.168.0.11
    Netmask         255.255.255.0
    Gateway         192.168.0.1

CHANNELS
    Number          1...30

HARDWARE
    Type:                   PMCprimo C    600MHz    CAN/CAN
    Mat.Nr.:                680055
    Ser.Nr.:                111
    Pr.Ver.:                1.0
    Inputs:                 6
    Outputs:                6
    Virtual-Inputs:         56
    Virtual-Outputs:        56
    Analogue Inputs:        0
    Analogue Outputs:       0

DEVICES in CAN Network:
    PMCprotego DS (S706) at CAN1 ADDR 5 found (DS402)
        Safety Card (DIS2) found: 680004 100001 1.0
        Inputs:             6 Inputs
        Outputs:            0
        Virtual-Inputs:     16
        Virtual-Outputs:    20
        Analogue  Inputs:   2   inputs  linked to channel from
0.1 to 0.2

    PMCprotego D (S706) at CAN1 ADDR 10 found (DS402)
        Inputs:             5
        Outputs:            1
        Virtual-Inputs:     17
        Virtual-Outputs:    19
        Analogue  Inputs:   2   inputs  linked to channel from
0.3 to 0.4
```

3.1 Data commands

n = 2: HW2 lists all CAN devices in table form. The state of devices, the firmware version number, the distinction between CAN 1 / CAN 2 and the link via PD and FS are listed.

```
STATE OF DEVICES:
Device            Network     Addr  CH    FS    VN          State
-------------------------------------------------------------------
PMCprimo C        ---         ---   ---   ---   3.0.2_T05   ACTIVE
PMCprotego DS     CAN1        5     0.2   27    5.260       FAULT &
                                                            WARNING
PMCprotego D      CAN1        10    0.3   27    5.260       ACTIVE
```

PR'Name' **Print to terminal**

(**Pr**int to terminal)

Permitted values for 'Name': Character string with maximum 80 characters.

This command lists the character string “Name” on a connected terminal. A total of 800 PR commands are possible within programs.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

SP[n] Save parameter

(**S**ave **p**arameter)

n = 0:	Everything is saved (does not need to be stated)
n = 1:	All program data is saved (excluding variables).
n = 2:	Only the variables are saved.
n = 3	All program data is saved (excluding Drive programs).
n = 4:	The data read out using the 'LA4' command in Motorola S-Record Format can be read in again using this option. This way it is possible to resave a complete backup. If a soft PLC is present, its data will also be read in again.
n = 5:	Saves all primo programs on the Protego's SD card
n = 6:	Saves all primo variables on the Protego's SD card
n = 7:	Saves the primo configuration on the Protego's SD card
n = 8:	Saves the soft PLC's boot project on the Protego's SD card
n = 9:	Saves the soft PLC's source code on the Protego's SD card

When saving to a SD card, the CD26 command (Page 3-17) must be used to select the PMCprotego containing the SD card.

Do not switch off during the save process, otherwise data will be lost.

Only the variables defined within a program will be saved.

3.1 Data commands

The time it will take to save the data depends on the amount of data to be saved. If there is insufficient memory available, the save process will be cancelled before the Flash memory is written. After power up, PMCprimo operates with the data stored within the Flash memory.

A check sum guarantees that the data has been read in correctly after start-up.

No programs should be running while data is being saved. The following procedure is recommended to ensure data is saved securely:

`GX`	Stop all programs.
`BV;BI`	Mask trigger variable and inputs.
`GF`	Switch off all motors
`SP`	Save data
`WT100;EV;EI`	Enable trigger variable and inputs after 100 ms.

The commands listed above are explained in detail further on in this manual.

All the drive parameters from the existing servo amplifiers are stored as programs. If necessary, the programs can be started in order to import a backup.

When the program is saved (not in the case of variables), a version number is incremented and is stored in the Flash memory. Program changes made by third parties can be recognised here, as the version number is changed when the program is saved. This number (#Version: XX) is listed with the “**LA**” command (Page 3-28). The version number is not listed if only the variables or the Flash memory are read (“**LA2**” and “**LA4**” command). If the application program is deleted via the boot menu or “**CD**” command (Page 3-17), the version counter is reset to zero at the same time.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

SU[n] Software update

(**S**oftware **U**pdate)

This command can be used to import new firmware. After you have entered the command, a confirmation prompt will appear. If this is confirmed, it will be possible to download the file containing the firmware in the Motorola S-Record Format, using the terminal program PTerm. The file has the extension "*.hex".

The new firmware will be active following a restart.

It is possible to start a software update of the SD card. All PMCtendos and PMCprotegos can also be updated as an option.
The firmware files on the SD card must be named in accordance with the following specifications:

Device	Name of firmware file
PMCprimo C	PMCprimoC.hex
PMCprotego D	PMCProtego.hex
PMCtendoDD4	PMCDD4.hex
PMCtendoDD5	PMCDD5.hex
PMCtendoDD5 mit Arm processor	PMCDD5N.hex

Example:

```
0.1: su
0.1:

Firmware update: (1=Yes/0=No/?=Help) ?

Please load the HEX file into the control system.
Then follow the instructions!

12566Programming...
Don't switch off now !!!

0.0 Software update successful (00:18:40)
0.1:
```

3.1 Data commands

The following updates are possible by entering a parameter:

SU0:	Firmware update, PMCprimo C classic
SU5:	Firmware update PMCtendo, all PMCtendo DD4
SU7:	Firmware update for all PMCtendoDD5
SU9:	Firmware update for all PMCprotegos
SU10:	Firmware update for all PMCprimo C
SU11:	Firmware update for all PMCtendoDD5 with ARM processor
SU13:	Firmware update for PMCprimoC via SD card of PMCprotego
SU14:	Firmware update for all PMCprotego via SD card of PMCprotego
SU15:	Firmware update for all PMCtendoDD5 via SD card of PMCprotego
SU16:	Firmware update for all PMCtendoDD5 with ARM processor via SD card of PMCprotego
SU 17:	Firmware update for all PMCtendoDD4 via SD card of PMCprotego

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

RS[n] Reset to default setup

(**Res**et to default setup)

This command overwrites all data with the factory-set data. This command is carried out automatically if a faulty check sum is detected during the boot process!

For n = 0 Above functionality applies.
For n = 1: Control system reboots.
For n = 2: PLC program is started
For n = 3: PLC program is stopped
For n = 4: PLC is reset and the PLC program is read in again from the Flash memory.
For n = 5 Reset PLC (cold)
For n = 6 Reset PLC (original)

RD[n] Reload stored data

(**R**eloa**d** stored data)

For n = 0: The stored data is reloaded by the internal Flash memory. All changes made since the last save operation are deleted in the process.
For n = 1: The stored PMCprimo programs are copied from the SD card into the internal Flash memory.
For n = 2: The stored PMCprimo variables are copied from the SD card of the PMCprotego into the internal Flash memory.
For n = 3: The stored PMCprimo configuration is copied from the SD card into the internal Flash memory.
For n = 4: The soft PLC's boot project is copied from the SD card into the internal Flash memory.
For n = 5: The soft PLC's source code is copied from the SD card into the internal Flash memory.

The SD card must be inserted into the SD card slot of a PMCprotego; the address of the PMCprotego should be stated via the command CD26. If the address 0 is stated, the system automatically searches for the first PMCprotego (lowest address) with SD card.

3.1 Data commands

LA[n] List all parameters

(**L**ist **a**ll parameters)

n = 0: Everything is listed (does not need to be stated)
n = 1: All program data is listed (excluding variables).
n = 2: Only the variables are listed.
n = 3: Flash data content is listed.
n = 4: Flash data content is listed in Motorola S-Record format.
n = 5: List is displayed during the boot process

This command is used to list all the data on the PMCprimo RAM. It can be used to carry out a full upload. The data received can be copied back into the control system via a download.

The LA command can be cancelled using the Escape key.

When n = 4, the flash data content is listed in a special format (S-Record). It can be read back in using the 'SP4' command. This way it is possible to save a program on the PMCprimo.

In the case of a servo amplifier, the drive parameters are also listed. They are stored in a program with a predefined name:

Name of program: DRIVE_PARA_n
whereby n: Address of network subscriber

If a soft PLC is used, its data will also be listed when n = 4.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

FM Display free memory

(Display **f**ree **m**emory)

This command displays the available memory capacity.

Example:

```
0.1: fm
0.1:
Free RAM memory 36491 of 53448 kBytes
Free Flash memory 3994419 of 4027056 kBytes
```

SK Set software licence key

(Set **s**oftware license **k**ey)

This command can be used to display the set software key and to set new software keys. The following software options are only available for an additional charge:

- Motion generator
- Soft PLC
- CNC functions

Where devices are ordered with the relevant options, the appropriate software keys are pre-enabled.

The “**SK**” command displays the unit’s serial number and the options that are currently enabled. It enables software keys to be entered and the current options to be disabled. A software option can be retrofitted by ordering a software key at a later date and entering it using the “**SK**” command.

Re-enter the relevant software key to disable an enabled option.

3.1 Data commands

This example deletes an installed software key:

```
0.1:
0.1: sk<CR>
0.1: Serial number: 000164
Installed software keys:
Motion: b976c5

New Key?    b976c5<CR>
Key: Entry successful!<CR>
0.1:
```


Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.1 Data commands

ZT[Name] **Display active programs**

(Display active programs)

This command lists all the programs active on the control system without stating a program name:

If a PLC program is active, it is displayed.

Example 1:

PMCprimo user	Comment
`0.1: zt`	Command is triggered
`0.1:`	
`Active processes:`	Current programs are listed:
`==========`	
`I0.1:3 => PRG1`	Program PRG1 is active and was called by input 0.1:3.
`ONLINE(1000) => PRG3`	Program PRG3 was called by an online command.
`$VAR1 => PRG2`	Program PRG2 was called by the trigger variable $VAR1.
`$VAR2 => PRG4 => PRG5`	Program PRG5 was called by program PRG4. Program PRG4 was called by the trigger variable $VAR2.
`0.1:`	

If the program name of an active program is also stated, the program line that is currently active will be listed.

Example 2:

PMCprimo user	Comment
`0.1: ZTPRG2`	"**ZT**" command with program name
`0.1:`	
`Active program: <PRG2> Subscriber: 0  Line: 2`	Line no. 2 of PRG2 is currently active.

3.2 Mode commands

Activate/deactivate position control loop

PC Enter position control mode

(Enter **p**osition **c**ontrol mode)

This command activates the motor's position control. Power is applied to the motor as a result.

Position control is deactivated if a fault occurs on the motor. PMCprimo signals active position control with the ">" character via Telnet. If a servo amplifier is to be enabled and it is positioned on a limit switch, a warning is issued:

"Controller warning: The NSTOP limit switch has been activated and the motor switched off! (468)" or
"Controller warning: The PSTOP limit switch has been activated and the motor switched off! (469)"

MO Motor OFF

(**M**otor **o**ff)

This command switches off the control enable. Power is no longer applied to the motor; the rotary encoder signals are still evaluated. If the control enable is re-activated, the motor remains in its current position.

The MO command can also be used as an alternative stop command. The GF command (see Page 3-71) can be used to execute the MO command simultaneously on all axes.

VMn Set virtual motor mode

(Set **v**irtual **m**otor mode)

Permitted values for n: 0 to 1

This command is used to declare an axis as a virtual axis. In virtual mode, the axis is restricted to generating the demand position; the actual position is compared with the demand position.

Application of virtual mode:

- Test functions and programs, without having a connection to the machine.
- Simulate a master axis.

VM0: The axis is a real axis. The rotary encoder input is the acknowledgement of the actual axis value.

VM1: The axis is a virtual axis. The axis does not enable the drive. Acknowledgement of the actual value is simulated internally.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

Basic commands for motor start and stop, using a normal, trapezoidal or sinusoidal motion profile.

MA±nn Move to absolute position

(**M**ove to **a**bsolute position)

Permitted values for n: ±2.000.000.000

The motor travels to the absolute position nn. The movement follows a trapezoidal or sinusoidal velocity profile (Fig. 01). Bit 2 of the CW command can be used to select whether acceleration is trapezoidal or sinusoidal. The motor accelerates with the acceleration value from “**SA**” (Page 3-45) and brakes with the braking ramp from “**DC**” (Page 3-46). The velocity is defined with the SV command (Page 3-44). Positional entries are made in increments. If you are using “MA” for round axes, see also SB command (Page 3-120).

It is possible to end a position mapping (XM command) with the MA command. Bit 6 from MW (Page 3-90) can be used to determine whether to travel to the specified position at the current velocity or at the velocity defined through “**SV**”. If Bit 2 from “**ZW**” is not set, the current cycle is maintained when decoupling. If necessary, the system is braked and travels to the required position in the opposite direction. If Bit 2 from “**ZW**” (see Page 3-44) is set, the direction defined with Bit 3 is always maintained. If this does not match the current direction, it will be changed.

When decoupling from a mapping in “**MA**”, the velocity is compared with “**SS**”. If the speed of the Slave is less than/equal to “**SS**”, then “**SS**” is used.

If Bit 1 of “**ZW**” is set, the shortest distance is always travelled.

3.3 Motion commands

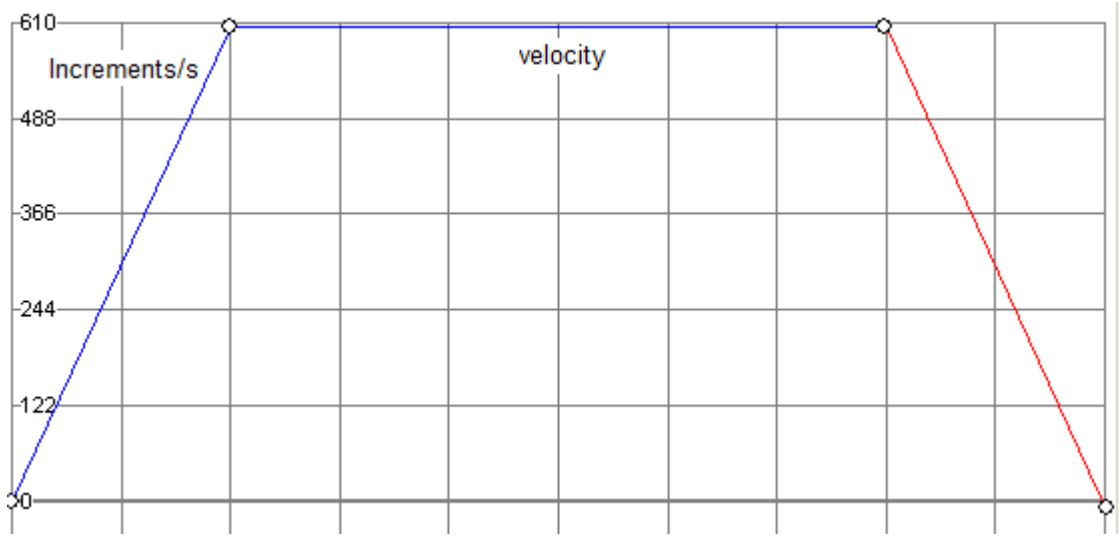


Figure 1: Trapezoidal velocity

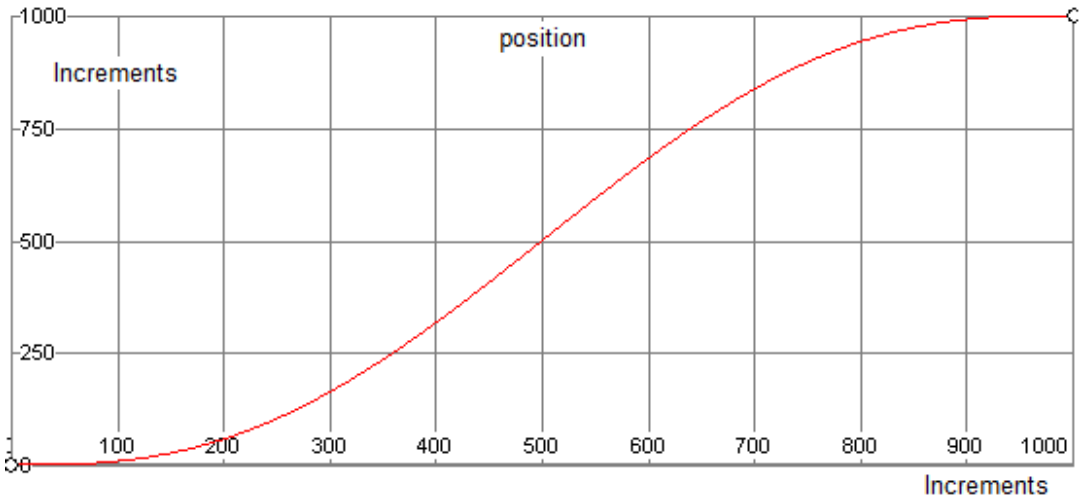


Figure 2: Course of position with trapezoidal velocity profile

If the distance to travel is short or the acceleration value is very low, the required velocity cannot be achieved. This produces a velocity profile that is triangular rather than trapezoidal (Figure 3).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

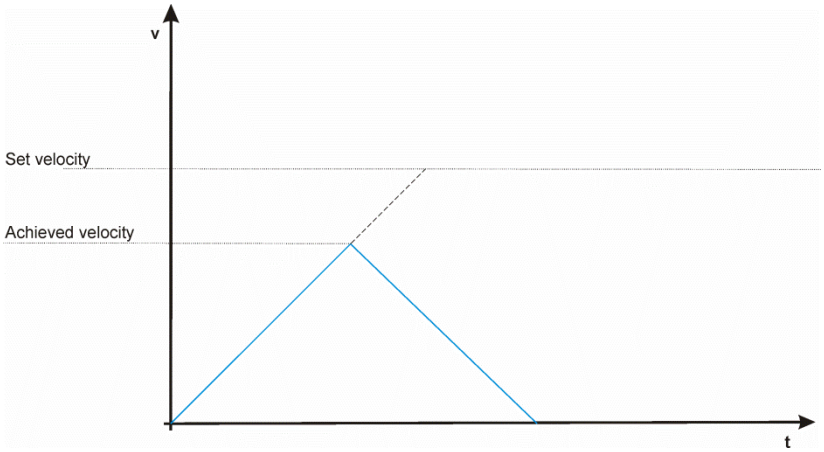


Figure 3: Triangular velocity profile

PMCprimo signals command execution with the “M” character via Telnet. Travel commands can only be executed when the position control loop is active.

Example 1:

PMCprimo user		Comment
0.1>	MA2000<CR>	The motor moves to the absolute position
0.1M		+ 2000 increments
0.1>		The motor has approached the position 2000

Example 2:

PMCprimo user		Comment
0.1>	MA5000;DP<CR>	Processing of the DP command begins
0.1M		as soon as the demand position has reached its target
0.1>DP=5000		
0.1>		

Before a travel command is executed, the target position is checked for compliance with the software end limits (LH and LL comand). If the target position is outside the software end limits, the travel command is not executed and PMCprimo signals the error “Target position is outside the software end limit”.

3.3 Motion commands

MR±nn Move units relative to current position

(**M**ove units **r**elative to current position)

Permitted values for n: ±2.000.000.000

The motor travels to the relative position nn. The motor moves nn increments from its current position. The movement follows a trapezoidal or sinusoidal velocity profile (Fig. 01).

The motor accelerates and brakes with the acceleration values that were specified with the SA or DC command. The velocity is defined with the SV command. Positional entries are made in increments. If you are using "**MR**" for round axes, see also SB command.

PMCprimo signals command execution with the "M" character. Travel commands can only be executed when the position control loop is active. Relative positioning is independent of Bits 1 - 3 of the MW command.

Example:

PMCprimo user		Comment
`0.1>`	`MR-3000<CR>`	The motor moves from its current position by 3000 increments in negative direction
`0.1M`		
`0.1>`		

While an MR command is being executed, the velocity value set using "**SV**" or "**SS**" can be changed. The speed level can be switched at any time using "**VJ**", but the value for "**SS**" must always be lower than the value of "**SV**".

Before executing a travel command, the target position is checked for compliance with the software end limits (LH and LL command). If the target position is outside the software end limits, the travel command is not executed and PMCprimo signals the error "Target position is outside the software end limit". The velocity during positioning does not change if the velocity is specified in a variable and the value of that variable changes during positioning.

It is possible to leave a position mapping (see XM command, Page 3-82) using the "**MR**" command.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

ST±nn Stop

(**St**op)

Permitted values for nn: ±2.000.000.000

If no parameter is stated, the motor will brake immediately to velocity 0 with the braking ramp specified via the DC command (Page 3-46). The ST command can be applied with any movement. PMCprimo signals “S” during the braking process.

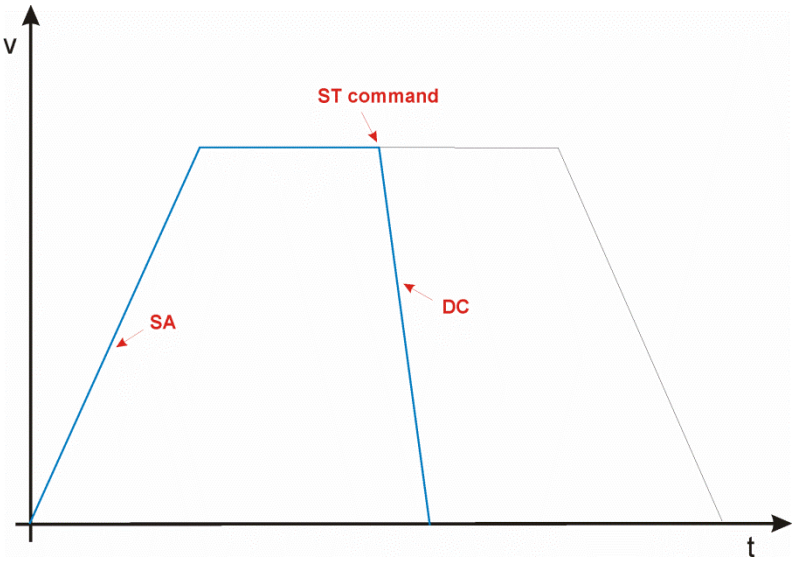


Figure 4: Movement ended via the ST command

If nn is not stated, the ST ramp is processed immediately.

If nn is stated, the system travels to the position nn. This is possible from any movement. During this time, PMCprimo signals “C”. This command can be used to decouple from a position mapping at an exact location, for example. If the specified position can never be achieved, PMCprimo remains in the “C” condition. This condition is then shown on the terminal.

3.3 Motion commands

If position mapping triggers a stop-at-position command (e.g.:“**ST0**”), the active curve is only maintained until braking has been started. The Slave is then stopped with a linear ramp (braking ramp depends on “**CT**” or “**CL**”). If the Master has also been stopped during this braking time, for example, the Slave will still move to the specified target position because the Master-Slave relationship is dissolved. In some circumstances, therefore, the Slave may travel a little further than specified by the Master.

However, Bit 6 of the “**MW**” command (Page 3-90) can be set so that the axis considers the curve even during the braking process and the Slave still reacts to the Master’s velocity changes. The Slave waits until the Master continues travelling, for example, and then completes the stopping process. Only the setting from the “**CL**” command is used, i.e. Bit 5 from the “**MW**” command (CT/CL selection) has no significance, as the target position cannot be reached with a time specification.

The ST command only affects the axis that is currently selected. The effect of the GS command (Page 3-70) spans more than one axis. With the AX command (Page 3-68), processing of a program can be cancelled without stopping the motor.

AB Abort, emergency stop

(**Ab**ort, emergency stop)

The motor brakes to velocity 0 with braking ramp XA (Page 3-46). The AB command can be applied with any movement.

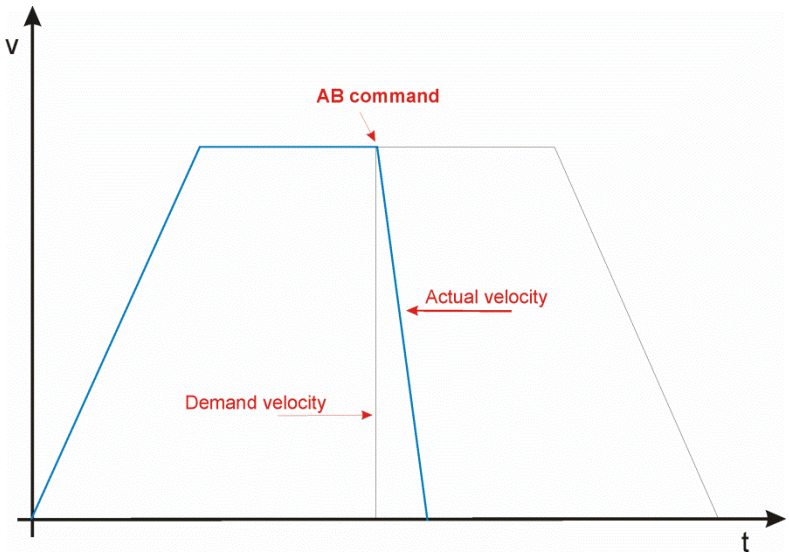


Figure 5: Movement ended via the AB command

The AB command only affects the axis that is currently selected. The GA command (Page 3-70) runs the AB command on all axes.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

VC[±] Move at constant velocity

(Move at **c**onstant **v**elocity)

This command is used to move the motor at constant velocity in the direction specified as + or -. If no sign is indicated, the motor will travel in the positive direction. PMCprimo accelerates the motor with the acceleration value SA (Page 3-45) until it reaches the velocity defined with the SV command (Page 3-44). The motor travels at constant velocity until it reaches a stop signal.

PMCprimo signals command execution with the “V” character at the Telnet interface. Travel commands can only be executed when the position control loop is closed.

Example:

PMCprimo user		Comment
`0.1>`	`SA1000;SV2000;VC+<CR>`	This command line sets the acceleration value to 1000 incr./sec.^{2}, the velocity to 2000 incr./sec.; it then accelerates axis 1 until it has reached the demand velocity and then travels at constant velocity.
`0.1V`		

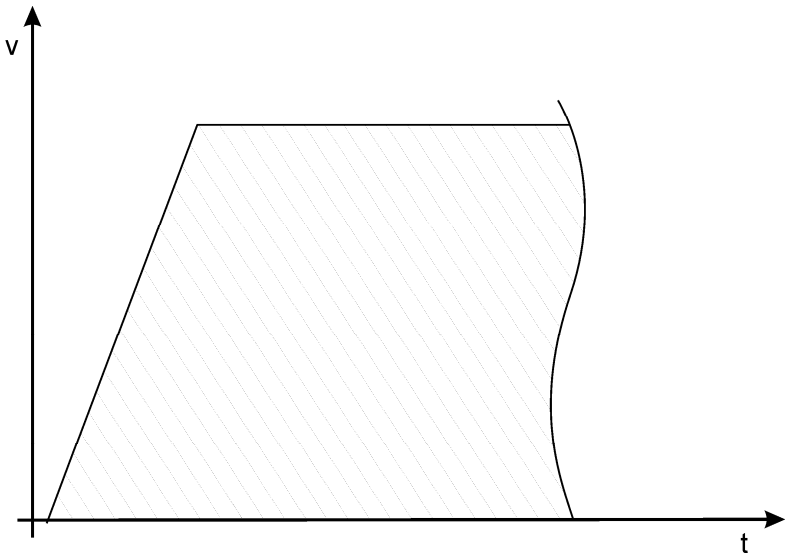


Figure 6: Moving at constant velocity

The velocity values SV and SS can also be changed at any time during travel.

If the axis is moving (pos. direction), the direction can be changed immediately by entering VC-.

3.3 Motion commands

IN[±] Initialise position

(**In**itialise position)

PMCprimo executes an initialisation command until a reference signal is detected. During initialisation PMCprimo signals “I”.

Execution: The motor accelerates to the demand velocity and then maintains this at a constant level until PMCprimo detects a reference signal. The direction of motor travel is specified by + or -. If neither + nor - is indicated, the motor will travel in the positive direction. The position counter is immediately set to RF (reference offset, Page 3-125) and the motor is braked. The motor then returns to Position 0.

The IN command for initialisation can also be used when the position control loop is open. In this case PMCprimo waits for a reference signal; when a reference signal is detected, the position counter is set to the value of “**RF**”.

The RW command (Page 3-122) can be used to prevent a return to the reference signal.

The RT command (monitoring rotary encoder signals, Page 3-109) is also effective when executing the IN command.

If initialisation is cancelled due to an error, the program that started the initialisation will also be ended.

Bits 2 and 3 of “**ZW**” (Page 3-56) are considered during initialisation. Bit 2 indicates whether the direction should be taken into account when returning to the reference signal. Bit 3 then indicates the direction when returning to the reference signal.

Example:

PMCprimo user		Comment
`0.1>`	`IN+<CR>`	The motor anticipates a reference signal in positive direction and then returns to the reference signal (to the new zero position when using “**RF**”).
`0.1>`		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

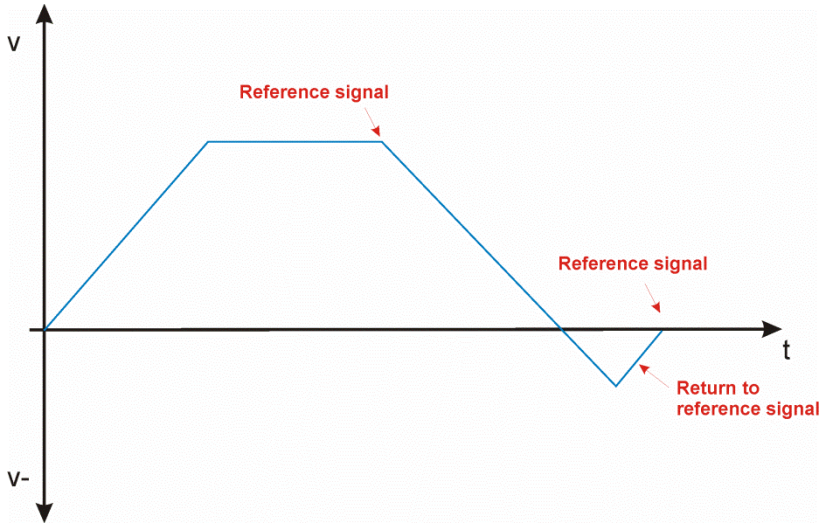


Figure 7: Initialisation with return to reference

3.3 Motion commands

IB[±] Initialisation position and bounds

(**I**nitialise position and **b**ounds)

The IB command is similar to the IN command. The IB command is used to initialise the position and also to determine the machine's cycle length. PMCprimo executes an initialisation command on the selected axis until a reference signal is detected for the first time. Then the axis continues to travel at constant velocity until the reference signal is detected for the second time. PMCprimo regards the gap between detection of the reference signals as the cycle length of the selected axis. During initialisation PMCprimo signals "I".

Execution: The motor accelerates to the demand velocity and then maintains this at a constant level until PMCprimo detects a reference signal for the second time. The direction of motor travel is specified by "**+**" or "**-**". If neither "**+**" nor "**-**" is indicated, the motor will travel in the positive direction. The position counter is immediately set to the value of "**RF**" (reference offset, Page 3-125) and the motor is braked. The motor then returns to Position 0.

The "**IB**" command for initialisation can also be used when the position control loop is open. In this case, PMCprimo waits for a reference signal twice. The gap between the reference signals is then entered in the SB parameter as the cycle length.

The RW command (Page 3-122) can be used to prevent a return to position 0.

"**IB**" may be assigned to a variable, e.g.: $VAR = IB. An "**IB**" is therefore executed and the result is stored in $VAR. "**SB**" is not changed in the process.

PMCprimo user		Comment
`0.1>`	`IB+<CR>`	The motor looks for 2 reference signals in positive direction and then returns to the reference signal (to the new zero position when using "**IB**").
`0.1>`		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.3 Motion commands

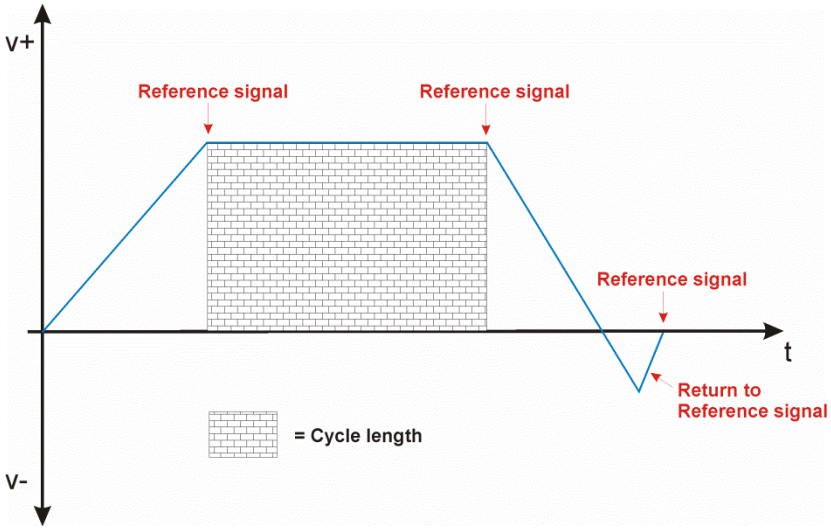


Figure 8: Initialisation of position and cycle length

CF Clear drive fault

(**C**lear drive **f**ault)

The CF command resets all existing faults and warnings on a servo amplifier.

If resetting the fault should trigger a restart of the servo amplifier, the system will automatically wait until the servo amplifier is once again ready for use.

3.4 Parameter commands

Input and output of axis parameters

SVnn Set velocity

(**S**et **v**elocity)

Permitted values for nn: 0 to 40,000,000
Factory setting: 20000

This command is used to specify the velocity in increments/second. The velocity can also be changed during travel.

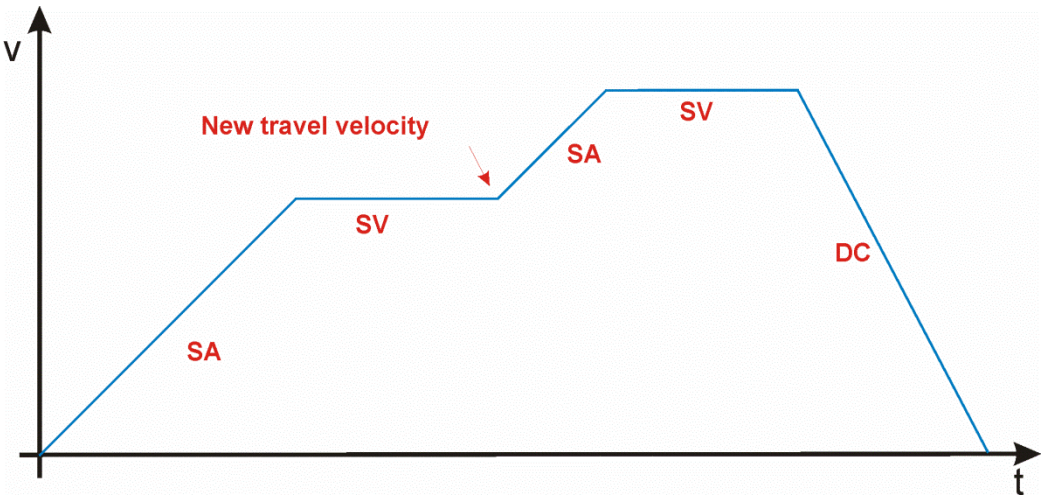


Figure 9: Changing travel velocity while moving

Example:

PMCprimo user	Comment
`0.1>  SV5000<CR>`	The velocity is set to 5000 increments/second.
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

SAnn Set acceleration

(**S**et **a**cceleration)

Permitted values for nn: 1,000 to 2,000,000,000
Factory setting: 100,000

This command is used to specify the acceleration value in increments/second^2. The acceleration value can be changed at any time, except while executing positioning tasks with “**MA**” or “**MR**” (Page 3-33, 3-36), i.e. including while moving. The lowest executable acceleration value is 1000.

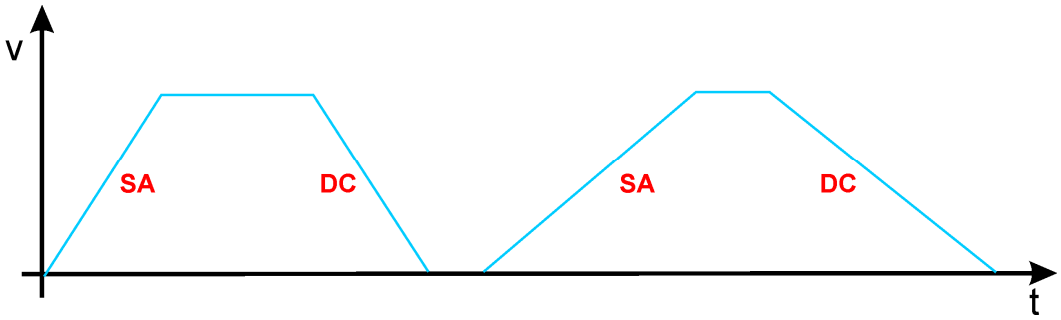


Figure 10: High and low acceleration value

Example:

PMCprimo user		Comment
`0.1>`	`SA10000<CR>`	The acceleration value is changed to 10000 increments per second^2.
`0.1>`		

3.4 Parameter commands

DCnn Set deceleration for ST command

(Set **dec**eleration for ST command)

Permitted values for nn: 1,000 to 2,000,000,000
Factory setting: 100 000

This command is used to specify the braking ramp in increments/second^2. The braking ramp can be changed at any time. The lowest, selectable braking ramp is 1000. The braking ramp is used with the "**ST**", "**MA**", "**MR**", "**IN**" and "**IB**" commands. "**XA**" can be used to set a separate braking ramp for the abort command "**AB**".

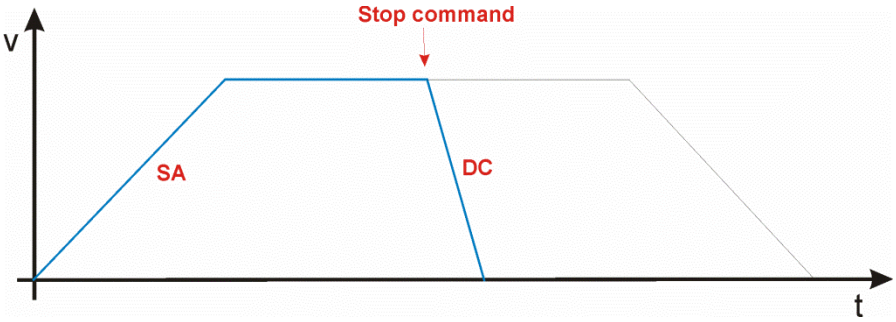


Figure 11: Set braking ramp

Example:

PMCprimo user		Comment
`0.1>`	`SA10000<CR>`	Acceleration 10,000 increments/second^2
`0.1>`	`DC25000<CR>`	Braking ramp of 25,000 increments/second^2 set at axis 1.

XAnn Set braking ramp for AB

(Set deceleration for AB command)

Permitted values for nn: 1000 to 2,000,000,000
Factory setting: 1,000,000

This command is used to specify the braking ramp for the abort command in increments/second^2. The braking ramp can be changed at any time. The lowest, selectable braking ramp is 1000. The "**XA**" braking ramp is only used with the "**AB**" command (Page 3-38).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

SCnn Set creep distance

(**S**et **c**reep distance)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

The normal trapezoidal velocity profile of a positioning movement can be supplemented by the introduction of a creep distance for positioning. The creep speed is set with the SS command. Creep mode is used to approach the target position slowly. This command is only effective if "**VJ**" (Page 3-49) is set to 0.

Example:

PMCprimo user		Comment
0.1>	SC200<CR>	Creep distance set to 200 increments
0.1>		

The braking process is now initiated so early that the axis travels at the slow creep speed at least 200 increments before reaching the target position.

3.4 Parameter commands

SSnn Set slow speed

(**S**et **s**low speed)

Permitted values for nn: 0 to 40,000,000
Factory setting: 0

This command is used to specify the creep speed in increments/second. The creep speed is the travel speed of the axis when "**VJ**" (Page 3-49) is set to 1. When executing a positioning command, the value for "**SS**" must be less than the value for "**SV**" (Page 3-44).

The SS speed is also used during a reference alignment run, when the axis should actually be at standstill.
It is also used in the alignment of "**MB**", "**MF**" and "**SM**" when the Master is stationary.

Example:

PMCprimo user	Comment
`0.1>  SS100<CR>`	Creep speed set at 100 incr./second.
`0.1>`	

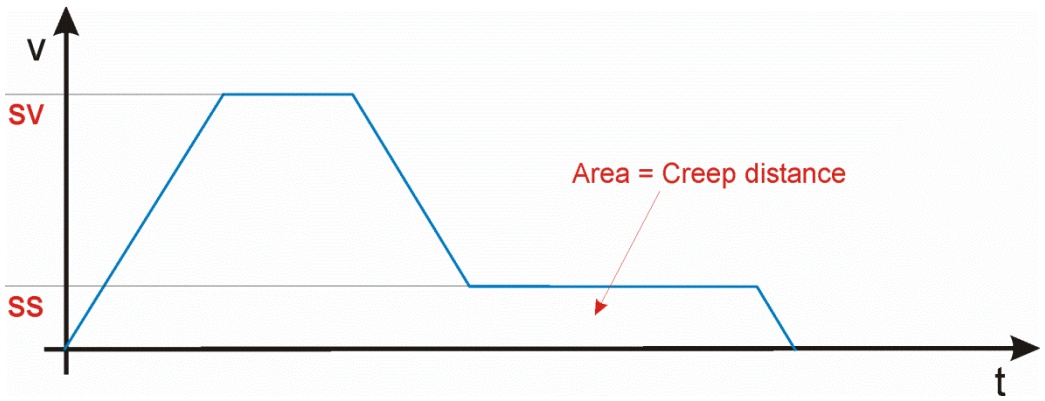


Figure 12: Positioning with creep speed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

VJn Set slow velocity mode

(Set slow velocity mode)

Permitted values for n: 0 and 1
Factory setting: 0

This command is used to switch between the velocity values “**SV**” and “**SS**”. When n=0, the axis travels at the velocity set using “**SV**” (Page 3-44). When n=1, the axis travels at the velocity set using “**SS**”. The setting for a creep distance (SC command, Page 3-47) is only valid when “**VJ**” equals 0.

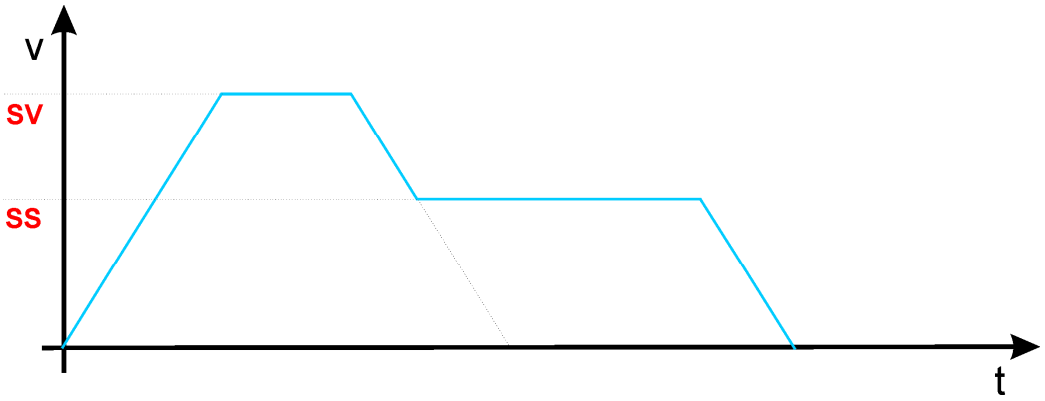


Figure 13: Velocity levels SV/SS

When switching between “**SV**” and “**SS**” during execution of “**MA**” or “**MR**”, the value specified for “**SS**” must always be less than the value for “**SV**”, otherwise the braking ramp cannot be calculated properly (braking ramp very steep).

3.4 Parameter commands

BLnn Set backlash compensation distance

(Set backlash compensation distance)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

This command enables mechanical backlash compensation. Each time the motor's direction of rotation changes, this stretch is added to the actual distance in defined increments. This backlash compensation is only effective when executing "**MA**" and "**MR**" commands.

Example:

PMCprimo user		Comment
`0.1>`	`BL20<CR>`	Define backlash compensation at 20 increments. Each time the direction of rotation changes, the first movement is increased by 20 increments
`0.1>`		

SWnn Set window

(**S**et **w**indow)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 100

This command is used to specify the permitted target window in increments. PMCprimo signals that the position has been reached when the motor is within the target window following the braking process. The target window is only used with the motion commands "**MA**" (Page 3-33) and "**MR**" (Page 3-36).

Example:

PMCprimo user		Comment
`0.1>`	`SW25<CR>`	The target window is defined at a size of 25 increments.
`0.1>`		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

TInn Set timeout for window

(Set timout for window)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

The target window “**SW**” is not examined until the time “**TI**” has elapsed. As a result, the “**MA**” or “**MR**” command is not completed until the time “**TI**” has elapsed.

Example:

PMCprimo user	Comment
`0.1>  TI200<CR>`	Monitoring time is set to 200.
`0.1>  MR200; MR200`	200 increments will be travelled twice.

Figure 14 illustrates the time “**TI**” that must elapse before “**MR**” starts for a second time.

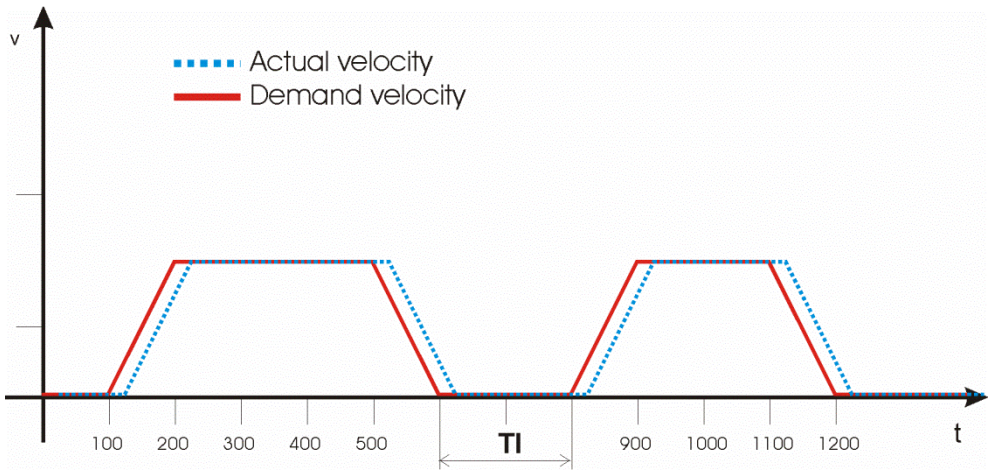


Figure 14: Monitoring time “TI”

3.4 Parameter commands

ISn Set increment select mode

(Set increment select mode)

Permitted values n: 0 to 8
Factory setting: 0

This command can be used to select a parameter which can be incremented using the IP command (Page 3-53). The parameter can be selected separately on each axis. Parameters from the following table can be selected:

n	Parameter	Limit values
0	No function	
1	Current velocity see below	
2	Reserved	
3	Position offset of master axis (MB)	±2,000,000,000
4	Position offset of slave axis (MF)	±2,000,000,000
5	Reserved	
6	Set travel velocity see below	
7	Transmission ratio (SMn/m)	0 – 400,000 for n
8	Cycle length (SB)	0 – 2,000,000,000

IS1: The lower limit for the current travel velocity is 0. If the axis is travelling at normal velocity, the upper limit is 2 x "**SV**". If the axis is travelling in creep mode, the upper limit is "**SV**". The incrementation only affects the current travel velocity, not the values for "**SV**" and "**SS**". The incremented value only applies until the next stop. All of the following movements are executed with the values from "**SV**" or "**SS**".

IS3/IS4: The incrementation of "**MB**" and "**MF**" changes the specified position offset. The AV command is effective when "**MB**" or "**MF**" is changed (Page 3-87, 3-88). The change is executed at the cycle limit.

IS6: The lower limit for the travel velocity "**SV**" is 0. If the axis is travelling at normal velocity, the upper limit is 2 x "**SV**". If the axis is travelling in creep mode, the upper limit is "**SV**". The incrementation not only affects the travel velocity but also the value of "**SS**" when travelling in creep mode, plus the value of "**SV**". All of the following movements are executed with the values of "**SS**" and "**SV**".

IS7: The lower limit for multiplication factor n of the transmission ratio "**SM**" (Page 3-88) is 0. The upper limit is 400000.

IS8: The lower limit for the cycle length "**SB**" (Page 3-120) is 0. The upper limit is 2,000,000,000.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

Example:

PMCprimo user		Comment
0.1>	SV1000<CR>	Normal velocity = 1000 increments/second
0.1>	IS1<CR>	Current speed can be changed via "**IP**"
0.1>	VC+	Travel at constant velocity
0.1V	IP100;DV<CR>	Increase velocity by 100 increments/second
DV1100		and display velocity
0.1V	IP100;DV	

IP±nn Increment selected parameter

(Increment selected parameter)

Permitted values for nn: Depends on selected parameter

This command increments the parameter selected using the IS command (Page 3-52) by the value stated with nn. "**nn**" may be either a positive or a negative value. The IP command enables the value of the travel velocity to be modified in stages by repeating a single command, for example. If the permitted limit value of the selected parameter is exceeded, an error message is issued.

PTnn Encoder filter

(encoder filter)

Permitted values for nn: 0 to 10,000

It is possible to average the encoder signals received over the period "**nn**". This can be used to smooth any "noisy" encoder signals that are received. This command can only be used on an uncontrolled axis (MO).

If a torque peak is present at a certain position on a master axis, this disturbance will also be seen on the slave axis. This filter can eliminate or reduce this effect.

Reaction to speed changes on the uncontrolled axis will be somewhat later due to averaging. They also lead to a static offset of the averaged position.

3.4 Parameter commands

CWbb Set control word

(Set control word)

Permitted values for bb: 0 to 1111 1111
Factory setting: 0001 0000

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

With this command the user can determine the encoder's count direction, the velocity profile for motion commands and the sign for the setpoint output of each axis (Bit 0 is on the right with input/output and on the left with Bit 7).

Note: The encoder's count direction and the sign for the setpoint output should only be changed when the position control loop is open. These inversion functions are used to simplify commissioning of the motor (saves on rewiring) and enable the travel direction of PMCprimo to be adapted to the user's definition of the travel direction.

Bit 0: Abort program in the event of a motor fault:
0: Axis programs that are waiting for a motion command to be run are aborted.
1: Axis programs are not aborted due to a motor fault.

Bit 1: Storing positions in non volatile memory (MRAM).
0: Positions are not stored in MRAM.
1: Current position is permanently held in MRAM. This bit cannot be stored. If the MRAM is to read in the last position on power-up, the bit must be set in the start-up program. The data is mirrored in the MRAM and checked on power-up. In the event of an error, an axis-specific error message is triggered.

Attention: This bit may only be set when all the commands that influence the position have been set ("**FS**", "**NB**", "**MS**", "**SB**" commands). If the bit is set too early, the position read from the RAM may be incorrect.

If the axis is in a negative position, it will still be negative after switching off/on. It's important to note that in the off state the axis may not be rotated manually by more than half the cycle length.

Bit 2: Definition of the velocity profile for acceleration and braking.
0: Trapezoidal velocity profile for the "**MA**", "**MR**", "**VC**" and "**ST**" commands
1: Sinusoidal velocity profile for the "**MA**", "**MR**", "**VC**" and "**ST**" commands

Bit 3: Behaviour in the event of a motor fault:
0: Axis is switched off immediately ("**MO**" command)
1: Axis brakes with the braking ramp "**AB**" and then switches off the motor.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Bit 4: Defines the sign for the setpoint output.

0: Setpoint sign is normal, i.e. if the encoder counts in a positive direction, the setpoint sign is negative.

1: Setpoint sign is inverted, i.e. if the encoder counts in a positive direction, the setpoint sign is also positive.

Bit 5: Defines the encoder's count direction.

0: Count direction normal, i.e. if Track B follows Track A, the position counter on PMCprimo counts in a positive direction.

1: Count direction inverted, i.e. if Track A follows Track B, the position counter on PMCprimo counts in a negative direction.

Bit 6: "**TO**" command for monitoring the encoder

0: "**TO**" command defines a monitoring time.

1: "**TO**" command defines a monitoring method.

Bit 7: Defines whether the KI factor is always active or only active at standstill.

0: The KI factor is always active.

1: The KI factor is only active when the drive is at standstill.

Example for Bit 0 of the "CW" command:

PMCprimo user	Comment
`0.1: CW1<CR>`	Axis program should not be aborted.
`0.1: PC;MR10000;WT5000;SV<CR>`	"**MR**" command triggers a motor fault.
`0.1>`	Status message for "**PC**"
`0.1M`	Status message for "**MR**"
`0.1 The position error limit has been exceeded! (01:48:28)`	
`0.1M`	
`0.1:`	
`0.1 SV=20000`	"**SV**" is listed after 5 seconds despite the error message.

3.4 Parameter commands

ZWbb Set position control word

(Set position control word)

Permitted values for bb: 0 to 1111 1111
Factory setting: 0100 0000

The wildcard 'X' can be stated for each bit. This means that this bit cannot be changed.

With this command the user can determine the behaviour of PMCprimo during positioning commands (Bit 0 is on the right with input/output and on the left with Bit 7).

Bit 0: Determines whether the position counter is signed.
0: Position counter negative and positive
1: Position counter positive only

Bit 1: Determines whether the cycle length influences positioning commands.
0: The cycle length set using "**SB**" (Page 3-120) does not influence the execution of positioning commands.
1: The cycle length is considered when executing absolute positioning (**MA**). With absolute positioning, the axis travels to the position within the cycle limit that corresponds to the absolute position (example: round axis does not travel 1.5 but only 0.5 machine cycles).

Bit 2: This bit specifies that the approach direction with absolute positioning (**MA**) is defined using Bit 3 of "**ZW**". This bit is also effective when Bit 1 is set.
0: Direction for approaching the demand position is not defined.
1: Direction for approaching the demand position can be defined with Bit 3.

Bit 3: This bit determines the approach direction of the demand position with absolute positioning on a round axis, if Bit 2 is set to 1.
0: Positioning is executed in a positive direction.
1: Positioning is executed in a negative direction.

Bit 4: If Bit 1 of "**ZW**" is set and the distance is a multiple of the cycle length, travel with absolute positioning (**MA**) depends on Bit 4:
0: If Bit 1 is set and distance=**SB**, travel will not occur
1: If Bit 1 is set and distance=**SB**, travel will occur

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.4 Parameter commands

Bit 5: This bit decides whether the cycle length "**SB**" should be considered for the **WA** command.

0: The cycle length **SB** is not considered for the **WA** command.

1: The cycle length **SB** is considered for the **WA** command, as follows:

Examples:
SB10000
WA12000 Waiting at position 2000 or –8000
(12000%10000 = 2000).
WA-12000 Waiting at position 8000 or -2000
(-12000%10000 + 10000 = 8000)

Standardising the wait command on the cycle length ensures that the command is completed during one cycle. The possibility of WA waiting indefinitely is excluded, e.g. if WA has been started with WA-12000 and the axis travels in a positive direction

Bit 6: Not assigned.

Bit 7: If the axis moves into mapping, this command can be used to correct the cycle length automatically. To do this, the cycle length specified using the SB command is multipled by the transmission ratio "**SM**".
As a result, a reference sensor is no longer necessary with uneven gear ratios.
Example: SB4096; SM1,3
The cycle length would be 1365.33. As a result, with a constant forward movement, a drift of one increment for every three pulses would arise. The cycle length is now set to 1365 for two pulses and to 1366 for one pulse, thereby avoiding the drift. The calculated cycle length can be checked using the "**GM**" command.

0: Automatic cycle length correction is inactive

1: Automatic cycle length correction is active

3.5 Program commands

Commands used to define and run programs.

Programs can be stored in the Flash memory.

Programs are composed of individual commands.

Commands on various axes can be executed in sequence or simultaneously.

In PMCprimo, all programs are stored on the host and are transferred from there to the necessary axes. A program consisting only of commands for one axis will be transferred in full to the respective axis module when called. The program remains in the axis module's local memory to be used again. Programs containing multi-host or multi-axis commands are analysed first by the host and are divided into their individual components, depending on whether the command needs to be executed by the individual axis or by the host.

Axis-specific program sections are loaded into the relevant axis module. When the program is executed, the host calls up the individual program sections in each axis module. The host numbers the individual program sections automatically. These are not accessible to users.

If a program is changed or deleted, the host changes or deletes the relevant local programs in turn.

The analysis and loading of programs (compilation) are automated. No input is needed from the user.

Commands and programs can be executed on various axes in sequence or simultaneously. Normally all commands are run in sequence, unless simultaneous processing has been selected using the "**CP**" command.

Example for processing in sequence:

PMCprimo user		Comment
0.1>	ES PRG1<CR>	Entry of PRG1
002E	CH1;MA20000<CR>	Axis 1 to absolute position 20,000
003E	CH2;MA10000<CR>	Axis 2 to absolute position 10,000
004E	NS<CR>	End of PRG1
0.1>		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

Example for simultaneous processing:

PMCprimo user		Comment
0.1>	ES PRG1<CR>	Entry of PRG1
002E	CP<CR>	Activate simultaneous processing
003E	CH1;MA20000<CR>	Axis 1 to absolute position 20,000
004E	CH2;MA10000<CR>	Axis 2 to absolute position 10,000
005E	CE<CR>	End simultaneous processing
006E	NS<CR>	End of PRG1
0.1>		

If PRG1 is started, the movement of axis 2 is started at the same time as the movement of axis 1.

In this example there is no connection between the two axes. If there is to be a link between the axes, this will need to be specified by the user.

One example for this is the point-to-point movement of two axes in an XY coordinate system. Both axes must have reached their positions before the next step can take place. The following example illustrates how the individual programs can be linked together.

Example for processing in sequence and simultaneously:

PMCprimo user		Comment
0.1>	ES PRG1<CR>	
002E	CP<CR>	Simultaneous processing on
003E	CH1;MA20000<CR>	Program PRG1: Axes 1 and 2 travel
004E	CH2;MA10000<CR>	to their absolute positions
005E	CE<CR>	Simultaneous processing off
006E	NS<CR>	
0.1>		
0.1>	ES PRG2<CR>	
002E	CP<CR>	Simultaneous processing on
003E	CH1;MA0<CR>	Program PRG2: Axes 1 and 2 travel
004E	CH2;MA0<CR>	to their zero positions
005E	CE<CR>	Simultaneous processing off
006E	NS<CR>	
0.1>		
0.1>	ES PRG3<CR>	
002E	XS PRG1;XS PRG2<CR>	Program PRG3: Program 1
003E	NS<CR>	is processed and then
0.1>		program 2 is processed

3.5 Program commands

ES[Name] **Enter sequence**

(**E**nter **s**equence)

This command enables the user to enter a program. Programs created using the ES command can be called and executed using the "**XS**" or "**XP**" commands. A program becomes a subprogram when it is called by another program.

The number of possible programs depends on the memory capacity of the integrated Flash memory. Each program can be called as a subprogram. It is not possible to have more than 30 levels for subprogram calls.

Each program line consists of a single command or a command string. Each program can consist of several program lines. Program entry is completed with the "**NS**" command.

A program name must always start with a letter. A program name can be a maximum of 20 characters in length. Spaces and special characters are not permitted.

Example 1 for program entry:

PMCprimo user		Comment
0.1>	ES PRG10<CR>	Entry of program PRG10
002E	IN-<CR>	Initialisation in minus direction
003E	RP3;MR100<CR>	Movement of 100 increments in + direction Execute command line 3 times
004E	MA2000<CR>	Movement to absolute position 2,000
005E	NS<CR>	Complete program entry
0.1>		Program entered

Example 2 for deleting a program:

PMCprimo user		Comment
0.1>	ES PRG10<CR>	Entry of program number 10
002E	NS<CR>	Entry complete
0.1>		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

NS End of sequence

(E**n**d of **s**equence)

This command completes the program entry. Each program must end with this command.

LOOP Loop

(loop)

The program name '**LOOP**' is reserved for a program with cyclical program processing. A program defined with "**LOOP**" will be run continuously with lower priority. The "**LOOP**" program is not started automatically on power-up, The "**XP**" command (Page 3-64) starts the program as a continuous loop. With the "**XS**" command (Page 3-63), the program is run once only. The "**GX**" (Page 3-71) or "**GX LOOP**" command ends the continuous loop. The "**ZT**" command (Page 3-30) indicates whether "**LOOP**" is active.

The following commands are not permitted within a "**LOOP**" program:

"**SP**": Save parameter (Page 3-22)
"**CP**": Change to parallel mode (Page 3-70)
"**DM**": Continuous display mode (Page 3-190)
"**TR**": Enable trace mode (Page 3-191)
"**CD**": Change operating mode (Page 3-17)
"**XM**": Execute map (Page 3-82)
"**XS**": Execute sequence (Page 3-63)
"**SK**": Set software licence key (Page 3-32)
"**LM**": List map (Page 3-81)
"**XX**": Length of alignment move (Page 3-84)
"**LB**": List bus variable value (Page 3-220)
"**LV**": List variable value (Seite 3-220)

3.5 Program commands

LS[Name] **List sequence**

(List sequence)

This command enables the user to list a program. If the "**LS**" command is triggered without a 'Name', all available programs will be listed. Details of whether the program is an axis program or a host program will also be listed. An axis program only contains axis-specific commands.

Example 1: List program PRG10

PMCprimo user		Comment
0.1>	LS PRG10<CR>	List program PRG10
	IN-	
	RP3;MR100	
	MA2000	
	NS	
0.1>		

Example 2: Display a list of all available programs in PMCprimo

PMCprimo user		Comment
0.1>	LS<CR>	List available programs
TEST	(NODE 0)	TEST program available
PRG10	(HOST)	PRG10 program available
PRG15	(NODE 0)	PRG15 program available
0.1>		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

XSName **Execute sequence**

(E**x**ecute **s**equence)

This command enables the user to call a program. When a program is called, the stated program is run immediately. If an error occurs, program processing is aborted immediately. Program processing can be ended at any time using the "**GX**" command (Page 3-71). A program must be run in its entirety before being called again (if this is disregarded, PMCprimo issues the following error message: "Cannot start the program <Name> because it is already running!"). Subprograms are executed consecutively.

Example: Call program PRG10

PMCprimo user		Comment
`0.1>`	`XS PRG10<CR>`	Call program PRG10
`0.1M`		PMCprimo executes the program PRG10.
`0.1>`		Program PRG10 has been executed.

3.5 Program commands

XPName **Execute sequence parallel**

(E**x**ecute sequence **p**arallel)

This command enables the user to call a program in parallel. Any subprogram called by a program will be started in parallel. The main program will continue running. If the program triggers a change of state at an axis, this will be displayed. If an error occurs, program processing is aborted immediately. Program processing can be ended at any time using the "**GX**" command (Page 3-71). A program must be run through in its entirety before being called up again (if this is disregarded, PMCprimo issues the following error message: "Cannot start the program <Name> because it is already running!").

Note: A maximum of 25 programs can be started in parallel.

Example: Parallel program call

PMCprimo user		Comment
0.1: ES PRG11		Program PRG11 is defined
002E	XP PRG12	Program PRG12 is to be called in parallel.
003E	SV	Velocity is listed.
004E	NS	End of definition of PRG11.
0.1:	ES PRG12	Program PRG12 is defined
002E	WT10000	Wait for 10 seconds.
003E	SA	The acceleration is then listed.
004E	NS	End of definition of PRG12.
0.1:	XS PRG11	Program PRG11 is called.
0.1:		
0.1	SV=20000	Velocity is listed immediately.
0.1	SA=100000	Acceleration is listed after 10 seconds.
0.1:		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

RP[nn] Repeat command line

(Repeat command line)

Permitted values for nn: 1 to 4,000,000, or no details

With this command, the current command line is repeated nn times (if "**nn**" is not stated, then infinitely). The "**RP**" command may only be used once in a command line. The "**ER**" command can be used to exit a loop when it ends. The AX command (Page 3-68) or GX command (Page 3-71) ends the loop immediately.

Example:

PMCprimo user		Comment
`0.1>`	`RP5;MA100;MA0<CR>`	Axis 1 to Pos. 100 /Axis 1 to Pos. 0
`0.1M`		The program line is executed 5 times.
`0.1>`		

ER End repeat

(End repeat)

This command can be used to exit a program loop programmed with the RP command at its end. The ER command is effective on continuous as well as non-continuous program loops (for example see IF command, Page 3-66).

3.5 Program commands

IFVerknüpfung Execute commands if expression is true

(Execute commands if expression is true)

This command can be used to execute a command or command line based on whether the result of an arithmetic expression is TRUE. If the operation is true, the commands that follow the expression will be executed to the end of the line or to the "**EL**" command. The operation that follows the IF command may be a simple comparison with a variable or an arithmetic expression.

Example 1:

PMCprimo user	Comment
`0.1> RP;IF($X<10);$X=($X+1);SV($X*10000);WT1000;EL;ER`	
`0.1>`	This command line increments the variable "**$X**", sets a new speed value and then waits for one second. If the value of the variable "**$X**" reaches the value 10, this command line has been processed in full.

Example 2:

PMCprimo user	Comment
`0.1>  IF($V1==2);XSPRG3;EL;XSPRG21<CR>`	
`0.1>`	With this command line, variable 1 is polled for the value 2. If the result is TRUE, PMCprimo calls program PRG3; if the result is FALSE it calls the program PRG21.

Example 3 for a more complex IF statement:

PMCprimo user	Comment
`0.1>  IF(($X>=10)&&($X<=20));SV($X*10000)`	
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

EL Else - execute commands if expression is false

(Else - execute commands if expression is false)

This command can be used to execute a command or command line based on whether the result of a preceding IF or II command (Page 3-66, 3-142) is **FALSE**. The EL command must be in the same line as the IF or II command.

Example:

PMCprimo user	Comment
`0.1>  IF(($B!=0)&&($A>=100));XS PRG_A;EL;XS PROG99`	
`0.1>`	This command line calls the program PRG_A only if the value of the variable "**$B**" does not equal zero and the value of the variable "**$A**" is greater than or equal to 100. Otherwise the program "**PROG99**" is called.

IXName Execute commands if program is active

(execute commands if program is active)

With this command, the commands that follow in the program line are only executed if the program 'Name' is active. If the program does not exist, an error message appears. The EL command (Page 3-67) can also be used in this case.

3.5 Program commands

AX Abort command execution

(Abort command execution)

This command interrupts the command that is currently being executed without stopping the motor.

If the AX command is used while the axis is travelling at constant velocity or a position mapping is activated, the axis will continue to travel at constant velocity and position mapping will remain activated. If the AX command is used while the axis is being positioned or is in a wait loop, the positioning or wait loop will be processed in full, but all subsequent commands will no longer be processed.

PMCprimo user	Comment
`0.1>  VC+<CR>`	Axis 1 travels at constant velocity
`0.1V  RP;DP;WT200<CR>`	Actual position is displayed every 200ms
`0.1V DP 1000`	
`0.1V DP 1500`	
`0.1V DP 2000`	
`0.1V DP 30000 AX<CR>`	Abort command execution
`0.1V`	Axes 1 travels at constant velocity

BK Break out of sequence

(Break out of sequence)

This command aborts the program that is currently being processed. This means that processing of the current program can be aborted when an input is polled, for example.

PMCprimo user	Comment
`0.1>  LS PRG10<CR>`	List program PRG10
`0.1>`	
`ES PRG10`	
`II0.1:1-;BK`	Abort program, if the state of input 0.1:1 is 0, otherwise travel with axis 1 to the absolute position 1000.
`CH1;MA1000`	
`NS`	
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

ASName **Set autostart sequence**

(Set autostart sequence)
Factory setting: 0

This command can be used to define a program as a start-up program in PMCprimo. This program is always processed after power is switched ON. If no program name is stated, the name of the current start-up program is listed. If 0 is entered as the program name, this means that the user does not require a start-up program.

CHn **Change channel**

(**C**hange c**h**annel)

Permitted values for n: 1 to 60 (can be set via CD command)

This command is used to select the required axis. An axis can be changed at any time.

Example:

PMCprimo user		Comment
`0.1>`	`CH1;MA200;CH.2;MA100<CR>`	Axis 1 to position 200 and then axis 2 to position 100.

3.5 Program commands

CP Change to parallel mode

(**C**hange to **p**arallel mode)

This command defines the start of simultaneous processing of axis commands. The "**CP**" command is only permitted within programs.

Example:

PMCprimo user		Comment
0.1>	ES PROG1<CR>	PROG1 is entered
002E	CP<CR>	Simultaneous processing on
003E	CH0.1;MA200<CR>	Axis 1 to travel to Pos. 200
004E	CH2;MA100<CR>	Axis 2 to travel to Pos. 100
005E	CE<CR>	Simultaneous processing off
006E	NS	End of program
1.1>		

The movements within the program "**PROG1**" will be started at the same time if the program is called.

CE End of parallel mode

(End of parallel mode)

This command ends simultaneous processing of axis commands. The "**CE**" command is only permitted within programs. If the "**CP**" command is not present before "**CE**" is entered, PMCprimo will signal "Select parallel mode first".

GS Global stop

(Global stop)

All axes brake with their respective braking ramp "**DC**". The GS command is an ST command (Page 3-37) that encompasses all axes.

GA Global abort

(Global abort)

All axes brake with their respective braking ramp "**AX**". The GA command is an AB command (Page 3-38) that encompasses all axes.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.5 Program commands

GF Global motor off

(Global motor off)

All motors are shut down. The GF command is an MO command (Page 3-32) that encompasses all axes.

GE Global wait off

(Global wait off)

All running wait loops are ended immediately. The GE command is a WE command (Page 3-105) that encompasses all wait loops.

GX[Name] Global abort execution

(Global abort execution)

This command sends the “Abort command” command to all axes and also ends programs at HOST level. If a command line or a program begins with the GX command, all running programs will be aborted; the commands following the GX command will then be executed. If a program name is stated, only the stated program will be aborted.

MEName Set motor off error sequence

(Set **m**otor off **e**rror sequence)

Permitted entry: Program name or 0
Factory setting: 0

This command can be used to define a program as a reaction program in PMCprimo in the event of an axis error. This program is always run when an axis error occurs. If no program name is stated, the name of the current reaction program is read in the event of an axis error. If 0 is entered as the program name, this means that the user does not require a reaction program if an axis error occurs.

A reaction program can be defined for each axis.

3.5 Program commands

The following error messages trigger a reaction program:

Code	Error
5	Software end limit has been passed in the plus direction!
6	Software end limit has been passed in the minus direction!
23	Insufficient RAM!
24	Position error limit has been exceeded!
29	No encoder signals received!
37	Reference error limit has been exceeded!
38	Reference error could not be corrected in one cycle!
39	The monitoring time for the reference signal has elapsed!
56	The limit switch has been activated and the motor switched off!
74	Monitoring time for master positions has elapsed!
117	Value for analogue input is below the lower limit value
118	Upper limit value for analogue input has been exceeded
141	Master value is not available in the position table
172	Controller error: Heat sink temperature > 80 degrees Celsius
173	Controller error: Overvoltage on intermediate circuit
174	Controller error: Open circuit, short circuit in feedback (Resolver, Hiperface)
175	Controller error: Undervoltage on intermediate circuit
176	Controller error: Motor limit temperature exceeded
177	Controller error: Problem with internal supply voltage
178	Controller error: Maximum speed exceeded
179	Controller error: EEprom check sum error
180	Controller error: Flash-Eprom check sum error
181	Controller error: Open circuit or short circuit on brake
182	Controller error: Motor phase missing
183	Controller error: Device's internal temperature too high
184	Controller error: Power level defective
185	Controller error: I2t limit exceeded
186	Controller error: 2 or 3 phases missing in the supply voltage
187	Controller error: A/D converter faulty
188	Controller error: Ballast resistance defective or faulty setting
189	Controller error: Phase missing in the supply voltage
190	Controller error: Problem with system software
230	No more reference positions can be buffered
249	Controller error: Commutation error

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

Creation and execution of tabular position mappings or electronic gear functions between axes in Master-Slave mode.

For each position of the master axis, a corresponding position is defined for the slave axis through tabular or linear position mapping.

The master axis can control a drive, but it is also possible for the master axis to operate only as a receiver of rotary encoder pulses (e.g.: transmission shaft, main shaft) or as a virtual axis (e.g.: speed-controlled stepping function).

When position mapping is activated, the corresponding position is assigned to the slave axis as the demand position from the table.

In simple cases it is sufficient to define a linear mapping as the electronic gearing between the master and slave axis. This electronic gearing function is available in PMCprimo as "**LINEAR**" position mapping. The master/slave transmission ratio can be set using the SM command.

Tabular position mapping is used for more complex applications. The more entries the table contains, the more precise the master/slave mapping will be, because PMCprimo carries out a linear interpolation between the individual table positions.

With tabular position mapping it is possible to reproduce mechanical gearing functions such as cam mechanisms or valve tappets on a camshaft.

3.6 Mapping commands

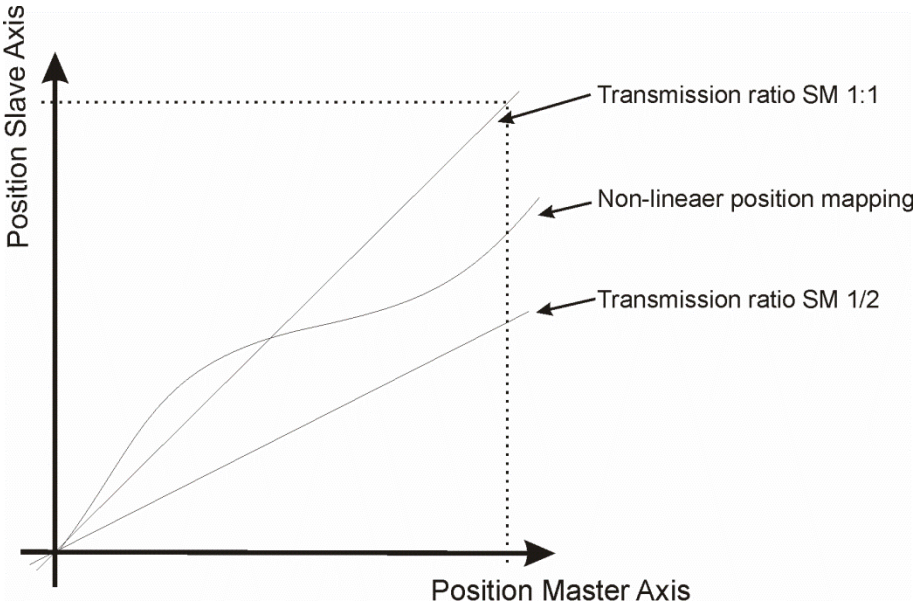


Figure 15: Simple position mappings

If the master and slave axis move within a specified area on a plant (e.g. XY table), position mapping need only be defined for this area. In this case it is not necessary to set a cycle length (factory setting of "**SB**" can be maintained), because the motors are not continuously travelling in the same direction.

The diagram below illustrates a typical, tabular position mapping. The slave axis is a cutting tool, for example, and the master axis is the material transportation. The speed of the master axis is to vary, so that the velocity of the material within the machine remains constant. By using tabular position mapping, the slave axis always travels to the corresponding demand position, irrespective of the current velocity of the master axis (the cutting tool always remains in the correct position for material transportation!).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

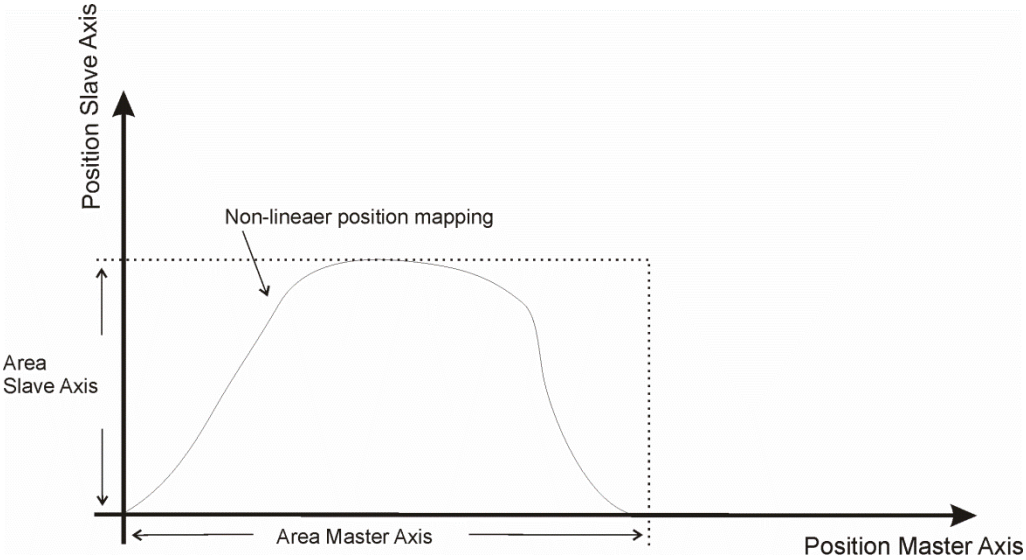


Figure 16: Position mapping for a defined area

The position mapping shown in Figure 16 can also refer to an application in which the master axis behaves as a transmission shaft and the slave axis follows the mapped position profile each time the transmission shaft (= main shaft) rotates. In this application, the SB command (Page 3-120) must be used to set the cycle length of the master to the measurement of one rotation of the transmission shaft.

If the master and/or slave axis behaves cyclically, the cycle length must be set on each axis. The tabular position mapping must be defined so that the position transitions run continuously at the cycle limits. If the transitions are not continuous at the cycle limits, erratic changes in speed may occur on the slave axis.

If the transition of the position values at the cycle limit of the slave axis is not continuous, the difference between the actual cycle length and the set cycle length causes a relative offset between the slave and master axis. This fault adds up over several machine cycles and behaves like a constant slave axis drift. This problem is difficult to diagnose, but it can be avoided by designing the tabular position mapping appropriately.

Tabular position mapping must cover the whole cycle length of the master axis from 0 to the cycle limit. The slave must know the cycle length of the master axis, so that it can calculate its speed and demand position beyond the cycle limit. This happens automatically when execution of a tabular position mapping is started via the ML command (Page 3-85). The cycle length of the master axis is automatically transferred to the corresponding slave axes.

3.6 Mapping commands

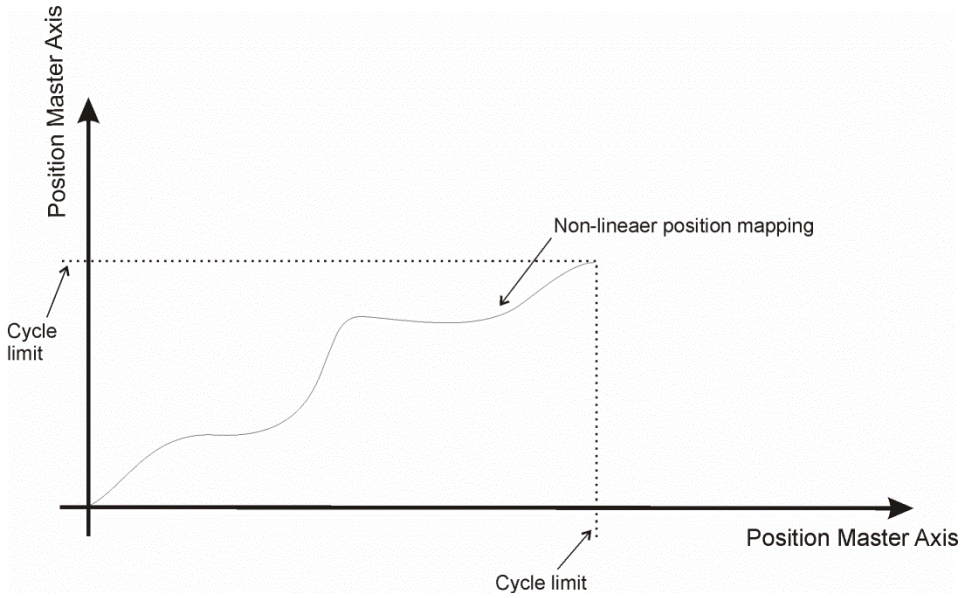


Figure 17: Position mapping for a cyclical machine

The diagram in Figure 17 illustrates a system in which the master and slave axes run their cycles at the same time with identical cycle lengths, although they move along different paths. It is not necessary for the cycle length and cycle time to match. With linear electronic gearing, neither the cycle length nor the cycle time match will match in practice. 1:1 gearing is an exception. The different cycle length and cycle time between master and slave axis does not cause any problem when tabular position mapping is executed. Neither is there any problem in exceeding the cycle length with the slave position when tabular position mapping is executed. In this case, the slave axis automatically compensates the zero point of its cycle length when the cycle limit is reached.

The position mapping between master and slave axis can be supplied with an offset value on both axes. These offset values (MB and MF command, Page 3-87, 3-88) move the master-slave relationship along the position axis of the master/slave. The value of the MB command is subtracted from the position of the master axis before the position is entered in the mapping. As a result, the position mapping line is moved to the right (see graphic below). The value of the MF command is added to the position of the slave axis before the position is entered in the mapping. As a result, the position mapping line is moved upwards. The “**MB**” and “**MF**” commands enable one slave axis, or all slave axes, to be moved in relation to the master axis with active position mapping.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

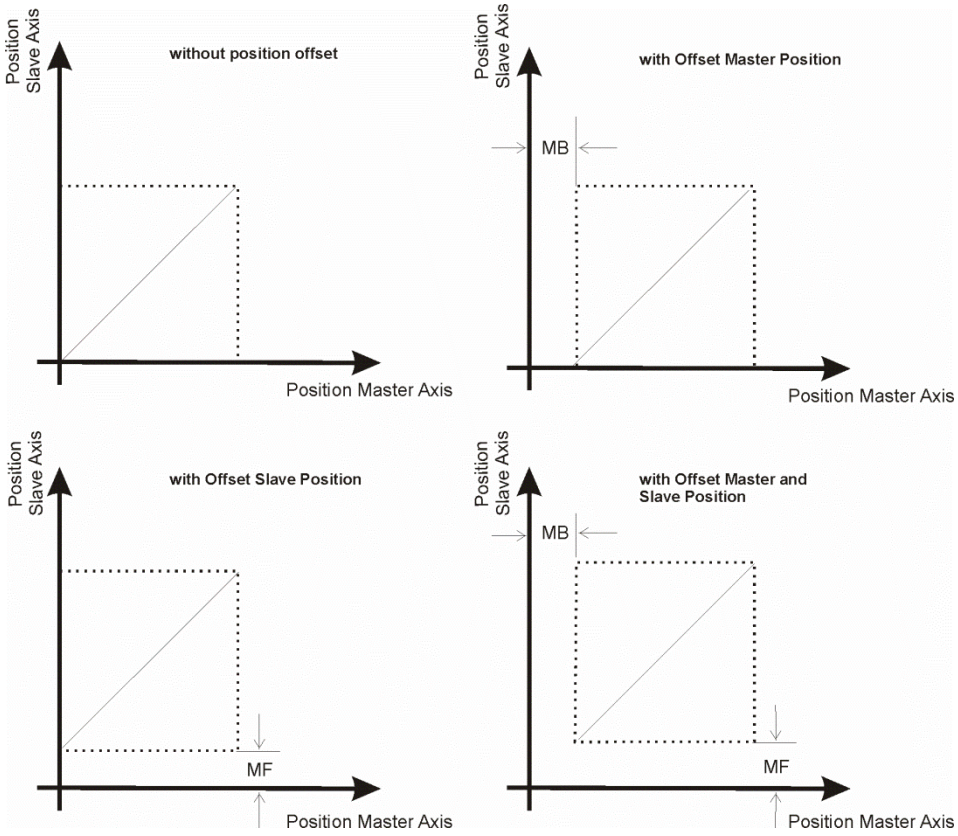


Figure 18: Effects of position offset

The position table data is entered as absolution slave axis positions. When absolute positions are entered for the slave axis, each value represents the demand position of the slave axis in reference to the master axis. The corresponding master positions result from the cycle length / number of positions.

The table positions are stored centrally. As a result, position mappings that are used in multiple axes need only be created once.

The position mapping data is transferred to the respective axis using the "**TM**" command (Page 3-85). The "**TM**" command must be executed once only at the respective axis, before an XM command is executed for the first time.

When position mappings are entered, the first position of the master is always 0. Only the entered table values are stored. The values between the table positions are determined through linear interpolation when the table increment is greater than 1. This enables the user to easily create a position mapping with relatively few table values over a large mapping area, without using much disk space.

Position mappings can be generated internally using the motion generator option, if this option is enabled using the SK command.

3.6 Mapping commands

The differential software function enables a drive to be synchronised so that it can follow the sum as well as the difference between the positions of two master drives. This simulates differential gearing. The “**ML**” command (Page 3-85) must be used to define the slave axis as the slave of a master axis and the “**NL**” command (Page 3-86) to define the slave axis as the slave of a second master axis (differential axis). Bits 4 - 6 of the LW command (Page 3-94) determines how the slave axis evaluates the positions of the master axes and Bit 0 of the LW command determines whether the master or the differential axis transmits its demand position or its actual position to the slave axis.

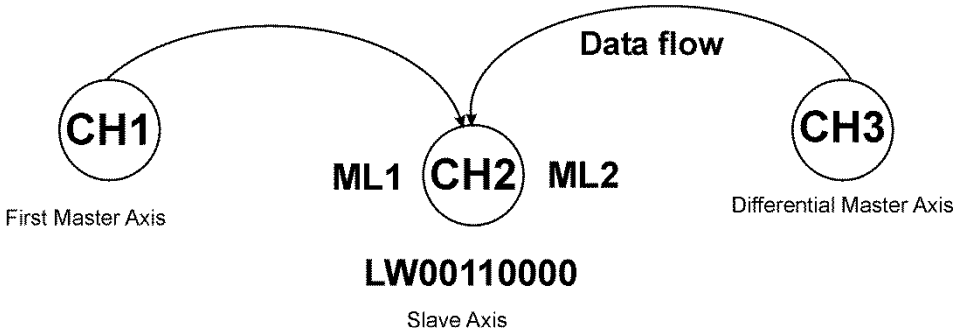


Figure 19: Example of software differential

The state of the master axes is arbitrary. It is important that the settings with the LW command (on the master and the slave axis) are made before executing the ML command. Changes to the LW command only take effect after the cancellation and redefinition as slave axis!

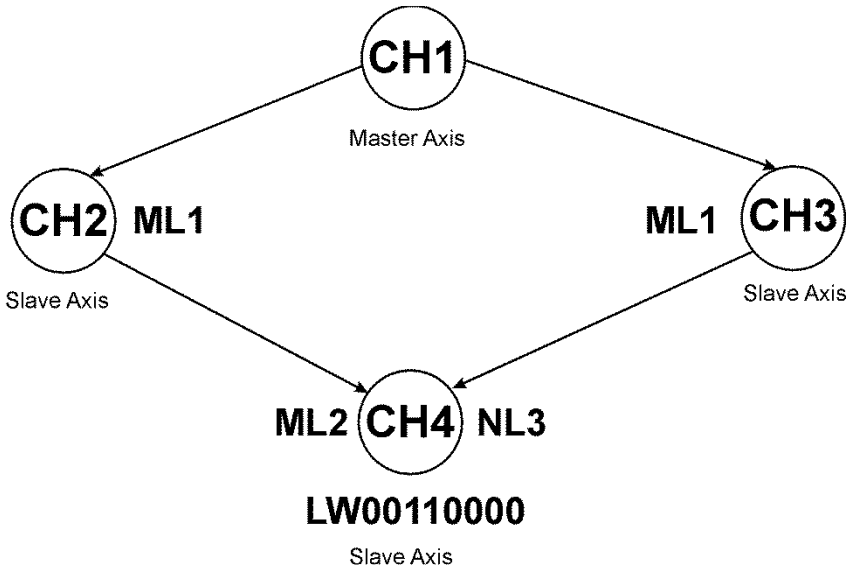


Figure 20: Example of software differential

The different parameters for executing a position mapping can be summarised in the equation below. All parameters are to be set on the slave axis, unless specific reference is made to the master axis.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

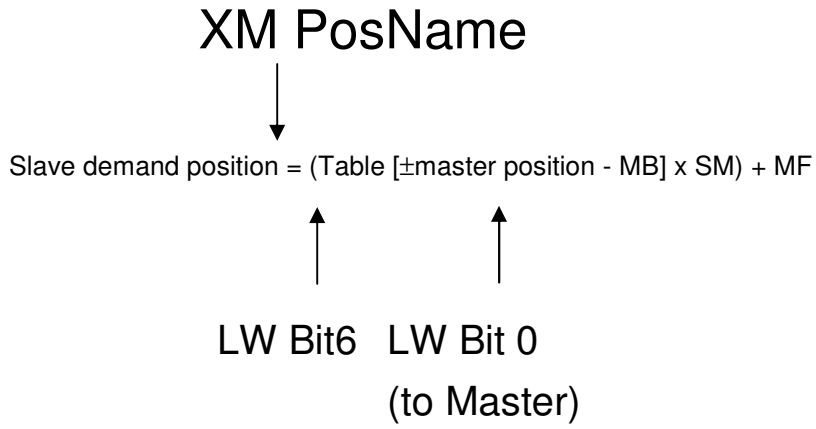


Figure 21: Position mapping as equation

$$S = \{Table\ [\ \pm(M_p \pm(M_d \times SB_{Master}/SB_{Diffmaster})) - MB] \times SM\} + MF$$

LW Bit5

Figure 22: Position mapping with differential as equation

S	Demand position of slave axis
M_p	Position of first master axis (ML)
M_d	Position of differential master axis (NL)
SB_{Master}	Cycle length of master axis
$SB_{Diffmaster}$	Cycle length of differential master axis

3.6 Mapping commands

EMName **Enter map**

(**E**nter **m**ap)

This command is used to enter position tables for position mappings.
A maximum of 500 tables can be created.

Example 1: Short position mapping with absolute slave positions

PMCprimo user	Comment
`0.1: EM TESTMAP<CR>`	Name of position table
`Cyclel`	
`Length of Master: 1000<CR>`	Master cycle length
`No. of values: 3<CR>`	Number of table values (in this case 3)
`001M 0<CR>`	Table value 1
`002M 500<CR>`	
`003M 1000<CR>`	
`004M NM<CR>`	End of table entry
`0.1:`	

Example 2: Deleting a position mapping.

PMCprimo user	Comment
`0.1>  EM Testmap<CR>`	Name of position table
`0.1>  Cycle length of Master: NM<CR>`	End of position table

NM **End of enter map**

(E**n**d of enter **m**ap)

This command ends the position table entry.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

LM[Name, n] **List map**

(**L**ist **m**ap)

Permitted values for n: 10000

This command is used to list a position table. If a table name is not stated, PMCprimo lists an overview of the position tables available within the memory.

Example:

PMCprimo user		Comment
0.1>	LM Testmap<CR>	List Testmap position table
0.1>	Testmap	Name of position table
1000		Master cycle length
3		Number of table values (in this case 3)
0		Table value 1 (must always be 0)
500		
1000		
NM		End of table entry
0.1>		

Exactly n values are listed with the optional parameter n. The values define the complete position table.

Example:

PMCprimo user		Comment
0.1>	LM Testmap<CR>,6	List testmap with 6 values
0.1>	Testmap	Name of position table
1000		Master cycle length
6		Number of table values (in this case 6)
0		Table value 1 (must always be 0)
200		
400		
600		
800		
1000		
NM		End of table entry
0.1>		

3.6 Mapping commands

XM[Name] **Execute map**

(E**x**ecute **m**ap)

This command is used to activate an existing position mapping. Activation takes place on the slave axis. Before activating the position mapping, the master-slave relationship must be defined using the "**ML**" command. Once the slave axis is connected to the master axis, the position of the master axis is transferred to the slave automatically by the host. An active position mapping is registered on the monitor with an "X" on the selected slave axis. The stop and abort commands deactivate a position mapping.

A linear position mapping is already pre-defined by the name **LINEAR**.

"**LINEAR**" position mapping is used specifically for executing transmission ratios specified using the SM command. Position mappings for linear correlations must not be generated using the EM command (Page 3-80).

If the command is triggered without a name, the position table that is currently active is listed.

Position mapping can be activated while the master axis is stationary or moving.

1. Activation when master axis is stationary:

In accordance with the activated position mapping, the slave axis moves at the velocity "**SV**" to the corresponding position, which has been mapped to the slave axis for the current position of the master axis. When software coupling is active (Bit 1 MW command, Page 3-90), the slave axis waits in the current position until the master axis is in the position that corresponds to position mapping and then engages. If the master is travelling at a velocity less than or equal to 500 incr/s, the slave engages even without software coupling. For this the velocity of the master should be averaged using the "**BT**" command (Page 3-202).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

2. Activation when master axis is moving:

Activation of position mapping while the master axis is moving must be executed with the help of software coupling (Bit 1 MW command).

When the XM command is used in a command line, all subsequent commands will only be executed once the slave axis is aligned or the slave access has engaged with the master axis.

Bits 4 and 5 of the AW command (Page 3-206) influence execution of the "**XM**" command when web tension control is activated with "**AM1**"

With velocity mapping (Bit 4 MW command, Page 3-90), only linear maps (XM LINEAR) are permitted.

"**XM**" may be assigned to a variable. In this case, the variable's current map number is transferred. If mapping is inactive, "**0xffff**" is transferred.

XVName, nn **Execute map virtual**

(e**x**ecute map **v**irtual)

Permitted values for nn: ±2,000,000,000

This command can be used to determine the slave position that corresponds to the master position 'nn' for the position table 'Name'. As a result, the slave position can be read in a variable, for example, without engaging. All settings ("**MB**", "**MF**" etc.) are considered in a similar way to "**XM**" (Page 3-82).

The position table must have already been transmitted to the axis using "**TM**". "**ML**" must have been executed on the axis.

Example:

PMCprimo user		Comment
0.1>	$VAR1=XV MAP_1,1000<CR>	Slave position is polled when using master position 1000 and map MAP_1.
0.1>	$VAR1<CR>	Value polled from $VAR1.
0.1>		
0.1>	$VAR1=2345 (0x00000929)	Slave position was stored in variable "**$VAR1**"

3.6 Mapping commands

XIName, nn **Execute inverse map virtual**

(e**x**ecute **i**nverse map virtual)

Permitted values for nn: -6,000,000 to 6,000,000

This command can be used to determine the master position that corresponds to the slave position 'nn' for the position table 'Name'. If several master positions are available, the first master position is always listed.

AI settings (“**MB**”, “**MF**” etc.) are considered.

The position table must have already been transmitted to the axis using “**TM**”. “**ML**” must have been executed on the axis.

Example:

PMCprimo user		Comment
0.1>	$VAR1=XI MAP_1,2345<CR>	Master position is polled when using slave position 2345 and map MAP_1.
0.1>	$VAR1<CR>	Value polled from $VAR1.
0.1>		
0.1>	$VAR1=1000 (0x00000929)	Slave position was stored in variable “**$VAR1**”

XXName **Length of alignment move**

(length of alignment move)

With this command it is possible to establish the necessary alignment run for the position table 'Name', which will be made automatically after starting the command “**XM** Name” (Bit 0 of “MW” (Page 3-90) must be set to 0). As a result it is possible to start the alignment run before executing “**XM**”. The “**XX**” command must always be executed on the slave axis.

The position table must have already been transmitted to the axis using “**TM**”. “**ML**” must have been executed on the axis.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

Example:

PMCprimo user	Comment
`0.1> ml2`	Axis 0.1 should be slave and axis 0.2 master
`0.1> zc0`	Slave position is set to 0.
`0.1> ch0.2`	
`0.1>`	
`0.2> zc300`	Master position is set to 300.
`0.2> ch1`	
`0.2>`	
`0.1> $XX=xx linear`	Result of XX is stored in variable $XX
`0.1> $XX`	Query value $XX
`0.1>`	
`0.1> $XX=300 (0x0000012C)`	Slave must travel 300 increments to arrive at the required position

TM[Name]

(**T**ransfer **m**ap data)

This command is used to send an existing position table from the host to the axis.

The position table itself is saved in the host's Flash memory and must be transferred back to the axis when necessary. This is the case, for example, when it has been recreated by the map generator.

If the command is triggered without a name, the position tables that have already been transferred are listed.

ML[n] Map link slave axis to master axis

(**M**ap **l**ink slave axis to master axis)

Permitted values for n: 1 to 60 (can be set via "**CD**" command)

This command is used to declare the current axis as a slave axis. n indicates the required master axis. This command must be used before activating a position mapping because PMCprimo cannot execute a position mapping if a master axis has not been defined.

3.6 Mapping commands

Example:

PMCprimo user		Comment
0.1>	CH0.2<CR>	Selection of axis number 2
0.2>	ML0.1<CR>	Axis 1 is master axis of axis 2
0.2>	CH0.3<CR>	Selection of axis number 3
0.3>	ML0.2<CR>	Axis 2 is master axis of axis 3
0.3>		

NL[n] Map link slave axis to differential master axis

(Map link slave axis to differential master axis)

Permitted values for n: 1 to 60 (can be set via "**CD**" command)

This command is used to declare the current axis as the slave axis of a 2nd master (differential). "**n**" indicates the required differential master axis. This command must be used before activating a position mapping. For details of the software differential please refer also to the LW command on Page 3-94.

UL Unlink slave axis from master axis

(**U**n**l**ink slave axis from master axis)

This command is used to cancel the mapping (link) to a master axis. The slave axis definition must be cancelled before the axis can be linked to another master axis as slave axis.

Example: Axis 1 becomes the new master axis

PMCprimo user		Comment
0.1>	CH3<CR>	Selection of axis number 3
0.3>	UL<CR>	Axis 3 is no longer slave axis
0.3>	ML1<CR>	Axis 1 is master axis of axis 3
0.3>		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

MB±nn Set map base offset for master map positions

(Set **m**ap **b**ase offset for master map positions)

Permitted values for n: ±4,000,000
Value range for n: ±2,000,000,000
Factory setting: 0

The MB command enables the introduction of an offset to influence the position mapping between master and slave axis. The value of the MB command is subtracted from the position of the master axis (this is the resulting position when using a software differential). As a result, the position mapping line is moved. The position mapping line normally runs between the zero point and the end point. If "**MB**" changes during a position mapping, an alignment occurs between the velocity defined via the "**AV**" command and the alignment acceleration "**AA**" (Page 3-89).

3.6 Mapping commands

MF±nn Set slave map position offset

(Set slave **m**ap position o**f**fset)

Permitted values for n: ±2,000,000,000
Factory setting: 0

The MF command enables the introduction of an offset to influence the position mapping between master and slave axis. The value of the MF command is added to the demand position of the slave axis. This enables the position mapping line to be moved in relation to the position of the master axis. The position mapping line normally runs between the zero point and the end point. If "**MF**" changes during a position mapping, an alignment occurs between the velocity defined via the "**AV**" command and the alignment acceleration "**AA**" (Page 3-89).

SMn,m Scale mapping

(**S**cale **m**apping)

Permitted values for n: 0 to 400,000
Permitted values for m: 1 to 400,000
Factory setting: 1.1

This command can be used to set the transmission ratio between master and slave axis. The transmission ratio is set at the slave axis. The absolute slave position is multiplied by "**n**" and divided by "**m**". If the transmission ratio changes during active position mapping, when the position at the slave axis is adjusted, the velocity for the alignment is determined with the "**AV**" parameter (Page 3-89) and the acceleration is determined with the value of the "**AA**" parameter (Page 3-89).

Example: Transmission ratio 1096/361
⇒ The actual demand position of the slave axis is the demand position from position mapping, multiplied by the factor 1096/361.

PMCprimo user		Comment
`0.1>`	`SM 1096,361<CR>`	The slave-master transmission ratio is set to 1096/361. Axis 1 is the slave axis in this case.
`0.1>`		

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

BR Set map scale factor from bounds ratio

(Set map scale factor from **b**ounds **r**atio)

This command can be used to automatically calculate the transmission ratio (SM command, Page 3-88) from the specified cycle lengths of the master and slave axis. The calculated transmission ratio equals the value "**SB**" (slave) / SB (master).

Example:

PMCprimo user		Comment
0.1>	SB4000<CR>	Cycle length 4000 increments axis 1
0.1>	CH2<CR>	Selection of axis 2
0.2>	SB5000<CR>	Cycle length slave axis 5000 increments
0.2>	BR<CR>	Calculate transmission ratio
0.2>	SM<CR>	List transmission ratio
SM=5000,4000		

AVn,[x] Set map base/offset/scale factor adjustment velocity

(Set map base/offset/scale factor **a**djustment **v**elocity)

Permitted values for n: 0 to 200%
Permitted values for x: 0 to 6
Factory setting: 0

The change/introduction of a position offset (MB and MF command, Page 3-87) or transmission ratio (SM command, Page 3-88) results in a new demand position for the slave axis. PMCprimo can use the AV factor to limit the maximum velocity during adjustment. When a position offset/transmission ratio is changed/introduced with an AV factor ≠ 0, a position change is not adjusted via one single dramatic setpoint change. The adjustment velocity is stated as a percentage, based on the current demand velocity. If the current demand velocity is 0, the adjustment is carried out using the velocity stored in the "**SS**" parameter.

Changing the transmission ratio when position mapping is activated on the slave axis also requires a change in velocity. In this case, the AV command operates in exactly the same way as when changing a position offset. The value for "**AV**" should be specified on the slave axis.

The "x" parameter can be used to increase the resolution to 0.1%, 0.01% etc.. In this case the velocity is multiplied by the factor 0.1^x.

Examples:
AV5.1 sets the adjustment velocity to 0.5%
AV5.2 to 0.05%
AV10.0 to 10%

3.6 Mapping commands

AAn Set map base/offset/scale factor adjustment acceleration

(Set map base/offset/scale factor **a**djustment **a**cceleration)

Permitted values for n: 0, 1000 to 2,000,000,000
Factory setting: 0

The change/introduction of a position offset (MB and MF command, Page 3-87) or transmission ratio (SM command, Page 3-88) results in a new demand position for the slave axis.
PMCprimo can use the "**AA**" parameter to determine the acceleration ramp for achieving the adjustment velocity. The "**AA**" parameter is stated in increments / second^2. The "**AA**" parameter should be set at the slave axis. If the parameter is 0, it is accelerated quickly. The lowest acceleration that can be set is 1000.

MWbb Set map options word

(Set **m**ap options **w**ord)

Permitted values for bb: 0 to 1111.1111. 1111.1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

This command is used to preset the behaviour of position mappings on the slave axis (Bit 0 is on the right with input/output and on the left with Bit 7):

Bit 0: This bit determines the behaviour of the slave axis when a position mapping is activated.

0: Software coupling is inactive, i.e., when a position mapping is activated, the slave axis travels to its assigned demand position.

1: Software coupling is active, i.e., the slave axis waits until the master axis assumes the corresponding position and then accelerates to the required velocity within the set time (CT command) or set distance (CL command), (see Bit 5), to comply with position mapping.
With velocity mapping (Bit 4 of MW command is set), position mapping can also be activated without software coupling. For this the master velocity must be lower than 500 increments/s.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

Bit 1: This bit determines whether a required alignment movement (e.g. after introducing MB and MF) should be dependent or independent of the set cycle length. Example: The axis position is -10000 increments and the cycle length is 10000 increments. If "**MB+20000**" is entered, an alignment movement of 0 increments results when Bit 1 is set to 1.

0: The cycle length has no influence on an alignment movement.

1: The cycle length is considered when executing an alignment movement. The shorter alignment route is always selected.

Bit 2: This bit determines whether the demand position is only approached in the direction selected with Bit 3 when a position mapping is activated.

0: Direction for approaching the demand position is not defined.

1: Direction for approaching the demand position can be defined with Bit 3.
If a start window is defined for the software coupling ("**CI**" command), this bit is ignored for the alignment run.

Bit 3: This bit determines the approach direction of the demand position when a position mapping is activated, if Bit 2 is set to 1.

0: The corrective movement is executed in a positive direction.

1: The corrective movement is executed in a negative direction.

Bit 4: Selection between position mapping and velocity mapping. Velocity mapping makes sense if absolute position mapping has no significance.

0: Position mapping

1: Velocity mapping
The BT command can be used to average the master velocity during velocity mapping. This will lessen any disturbances on the master.
It is possible to end a velocity mapping by stopping in position. Bit 6 of the MW command is ignored in this case.
The alignment acceleration "**AA**" is also active during velocity mapping if the transmission ratio is changed with "**SM**".

3.6 Mapping commands

Bit 5: Coupling can be executed over a preset time or a preset coupling length.
0: Coupling is executed with the set coupling time CT (Page 3-96).
1: Coupling is executed with the set coupling length CL (Page 3-96).
If "**ST**" is used, the map is still active when braking.

Bit 6: It is possible to disengage using "**MA**" or "**MR**". The velocity can be selected:
0: Disengaging with "**MA**" / "**MR**" occurs at the current slave velocity.1: Disengaging with "**MA**" / "**MR**" occurs at the velocity determined by "**SV**".
Disengaging with "**ST**" or "**ST**n" occurs in accordance with the MAP that will be exited. The gradient of the braking ramp is determined by the CL parameter, even if Bit 5 is set to 0.

Bit 7: Activate automatic cycle length correction. The length of the slave cycle is specified automatically, based on the position of the slave on the master cycle limit and its transmission ratio. The "**GM**" command (Page 3-189) can be used to list the current cycle length.

If the axis moves into mapping, this command can be used to correct the cycle length automatically. To do this, the cycle length specified using the SB command is multipled by the transmission ratio "**SM**".
As a result, a reference sensor is no longer necessary with uneven gear ratios.
Example: SB4096; SM1,3 (4096 / 3)
The cycle length would be 1365.33. As a result, a drift of one increment for every three pulses would arise with a constant forward movement. The cycle length is now set to 1365 for two pulses and to 1366 for one pulse, thereby avoiding the drift.
0: Automatic cycle length correction is disabled.
1: Automatic cycle length correction enabled.

Bit 8: This bit can be used to select whether to use the current SB value of the master axis or the master SB value from the position table when calculating the setpoint for the slave axis.
The SB value from the position table should be used if the master's cycle length is changed while the map is active.
0: When calculating the setpoint for the slave axis, the master cycle length is the current SB value of the master axis.
1: When calculating the setpoint for the slave axis, the master cycle length is the value from the position table.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Bit 9: Enable automatic map switching. If this bit is set, a new XM command can be triggered while mapping is enabled, without the map being immediately switched. Only when the master reaches its cycle limit does an automatic switch to the other map occur, if one has been defined.

0: Automatic map switching at master cycle limit disabled

1: Automatic map switching at master cycle limit enabled

Bit 10..15: Not assigned.

.

Special case “**MW101**”:
If Bit 0 and Bit 2 are set simultaneously, the following behaviour is defined:
The slave engages immediately. This results in a misalignment, which is saved as an offset for the slave in the “**MF**” parameter (Page 3-88). The offset can be compensated later through “**MF0**”.

3.6 Mapping commands

LWbb Set map link options word

(Set map link options word)

Permitted values for bb: 0 to 1111.1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

This command is used to set the axis coupling between the slave and master axis. The LW command is set on the slave axis; the setting must be made before the definition as slave axis via the ML command (Page 3-85) (Bit 0 is on the right with input/output and on the left with Bit 7).

With preset axis coupling it is possible to enable and define the software differential function. If the function is enabled, the demand position is calculated based on the sum or the difference between two master axes. The "**NL**" command (Page 3-86) must be used to define the 2nd master axis as differential master. The function can be enabled and defined with Bits 4 to 6.

Example:

PMCprimo user	Comment
`0.1: LW<CR>`	Axis 1 is the slave axis.
`0.1 LW=00000000`	
`0.1: LW00110000<CR>`	Enable software differential with subtraction
`0.1: ML2<CR>`	Axis 2 is the 1st master axis of axis 1
`0.1: CH2<CR>`	Select axis 2
`0.2: NL3<CR>`	Axis 3 is the 1st master axis of axis 2
`0.2: CH1<CR>`	Select axis 1
`0.1: PC<CR>`	Activate position control loop
`0.1> XM LINEAR<CR>`	After activating a position mapping on the slave axis, its demand position is calculated from the positions of the master axes, irrespective of their status in accordance with Bit 5 and 6. (Position mapping between axis 2 and 3 does not need to be present or active.)
`1X`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

Bit 0: This bit determines whether the slave axis is to follow the set values or the actual values of the master axis.

0: The slave axis follows the demand positions of the master axis.

1: The slave axis follows the actual positions of the master axis.

Bit 1: This bit determines whether the reference error correction is to be executed on the master or the slave. The bit must be set on the master axis in order to enable the function.

0: The reference error correction is executed on the master (if physically present).

1: The reference error correction is executed on the slave.

Bit 2: With velocity mapping it is possible to limit the slave velocity to a maximum (value of "**SV**") and a minimum (value of "**SS**"). Bits 2 and 3 of MW are considered here in order to define the direction.

0: Velocity restriction function disabled if Bit 4 of "**MW**" is set to 1.

1: Velocity restriction function enabled if Bit 4 of "**MW**" is set to 1.

Bit 3: It is possible to engage while the master is moving, without software coupling. However, this only makes sense if the moving master specifies standstill for the slave, as otherwise the slave would drop out with a position error.

0: Slave may not engage without software coupling while the master is moving.

1: Slave may always engage while the master is moving.

Bit 4: This bit enables the software differential function. The sum or the difference between two independent master positions determines the demand position of the slave axis.

0: Software differential function is disabled.

1: Software differential function is enabled.

Bit 5: This bit determines whether the demand position of the slave axis for the software differential function is the the sum or the difference between the master positions.

0: The master positions are added.

1: The position of the 2nd master axis is subtracted from the position of the 1st master axis.

Bit 6: This bit determines whether the sign for the demand position of the slave axis for the software differential function is negated.

0: The sign is not negated.

1: The sign is negated.

Bit 7: If velocity mapping is active (Bit 4 of "**MW**" is set), the acceleration and braking ramp can be restricted to the maximum values "**SA**" or "**DC**":

0: Restriction inactive

1: Restriction active

3.6 Mapping commands

CTn Set clutch time

(Set **c**lutch **t**ime)

Permitted values for n: 1 to 20,000
Factory setting: 1

The software clutch function is designed for connecting the slave axis to a moving master axis. The software clutch operates as a mechanical fixed point coupling, which engages with exact precision while moving. The slave axis waits in the current position until the master axis reaches the synchronisation position.

The software clutch time is only active if Bit 5 of "**MW**" (Page 3-90) is set to 0. If Bit 5 is set to 1, the software clutch length ("**CL**" Page 3-96) is active. This command defines the time for the acceleration ramp. The software clutch function is activated with Bit 0 of the MW command.

The value n=20000 represents the longest clutch time and the value n=1 the shortest. The clutch time t is stated in milliseconds.

Example:

PMCprimo user	Comment
`0.1>  CT16<CR>`	Specify software clutch time of 16 ms
`0.1>`	

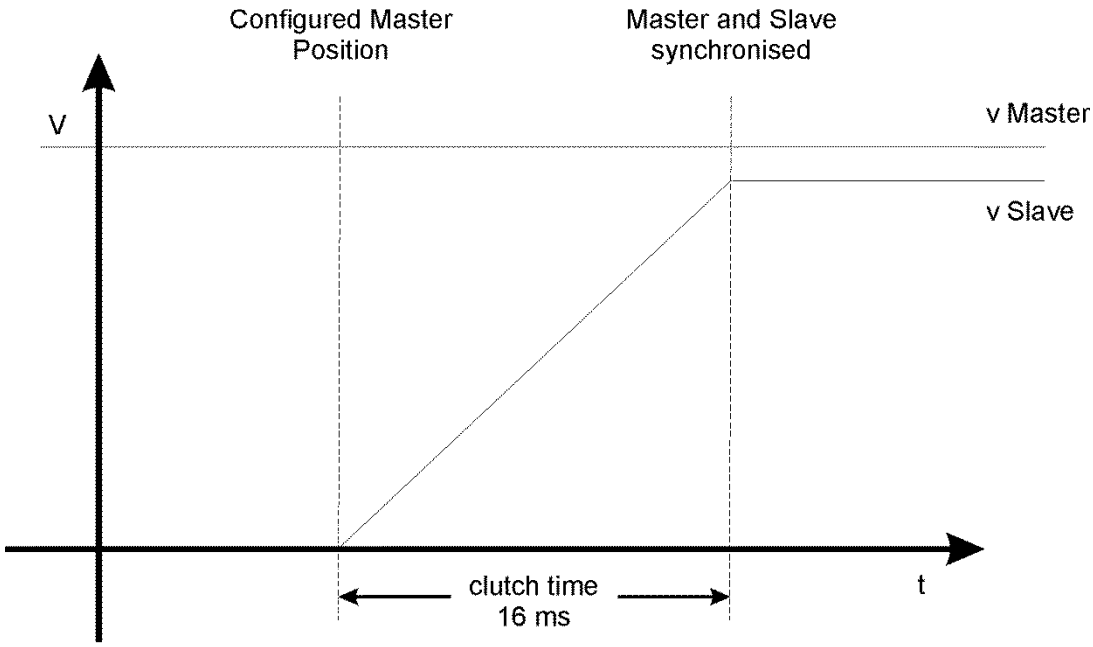


Figure 23: Software clutch behaviour

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.6 Mapping commands

CLn Set clutch length

(Set clutch length)

Permitted values for n: 0 to 2,000,000,000
Factory setting: 0

The software clutch length is only active if Bit 5 of "**MW**" (Page 3-90) is set to 1. If Bit 5 is set to 0, the software clutch time ("**CT**" Page 3-96) is active. This command states the acceleration ramp as a distance for the software clutch function. The software clutch function is activated with Bit 0 of the MW command. The software clutch function is designed for connecting the slave axis to a moving master axis. The software clutch operates as a mechanical fixed point coupling, which engages with exact precision while moving. The slave axis waits in the current position until the master axis reaches the synchronisation position. For speed-synchronous operation, a value of n≠0 should be specified for "**CL**" for connecting to the running master on the slave axis.

The value "**n=2,000,000,000**" represents the longest possible distance and the value "**n=0**" the shortest possible distance as clutch length. The clutch length is specified on the slave axis, but the stated distance in increments refers to the master axis.

Example:

PMCprimo user		Comment
`0.1>`	`CT16<CR>`	Specify software clutch time of 16 ms
`0.1>`		

3.6 Mapping commands

Cln Set clutch window

(Set **c**lutch w**i**ndow)

Permitted values for nn: 0 to 2,000,000,000

This command defines a start window for the software clutch.

If the master has stopped and "**Cl**" does not equal zero, an alignment run is always made based on the slave's starting position when the offset path < n. Bit 0 of the "**MW**" command (Page 3-90) is ignored in this case.
If the offset path > n, a software clutch is used to engage if Bit 0 of the **MW** command is set. If Bit 0 is not set, an alignment run is also carried out.

Bits 2 and 3 of "**MW**" are considered for the selection criterion.

If the master is moving, "**Cl**" is ignored and Bit 0 of MW again comes into force. The "**Cl**" command cannot be used with velocity mapping (Bit 4 of "**Cl**") and web tension control.

Bit 2 and Bit 3 of "**MW**" are not considered for the selection criterion.

MTn Master timeout

(Master timeout)

Permitted values for n: 0 to 1000

The "**MT**" command can be used to set a monitoring time to monitor the transfer of the master positions via the CANopen network. If no master values are received within the set time, an error message is issued and the motor is shut down.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.7 Wait loops

Wait loops can be programmed within a program or command sequence; PMCprimo signals “W” to the terminal program PTerm while a wait loop is active on the selected axis. The “**WT**” command is a host command. The number of potential WTs that can run simultaneously is unlimited. As a result, the message ‘W’ cannot be output for an individual axis.

WTnn **Wait for time**

(**W**ait for **t**ime)

Permitted values for n: 0 to 65535

This command can be used to program a time delay. PMCprimo waits for the time set under milliseconds to elapse before executing the next command.

Example:
Time delay of 2 seconds

PMCprimo user		Comment
`0.1>`	`MA2000;WT2000;MA0<CR>`	The motor moves to the absolute position +2000 increments, waits 2 seconds and returns to the absolute position 0.
`0.1M`		
`..1M`		
`0.1>`		

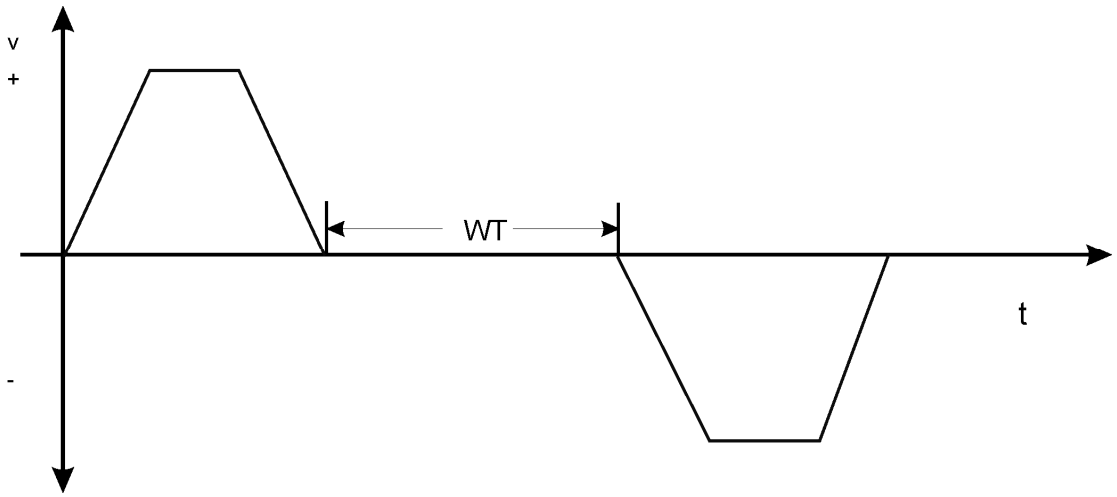


Figure 24: Time delay

3.7 Wait loops

WI[t.g:]n± Wait for input line n in group g

(**W**ait for **i**nput line n in group g)

Permitted values for t: 0 to 150 (depending on the address of the network subscriber) or no details

Permitted values for g: 1 to 8 (***PMCprimo*** C) 2
Permitted values for n: 1 to 8

This command can be used to wait for a positive/negative state of an input. PMCprimo waits for the relevant input state to be detected before executing the next command. If the byte address "**g**" is not stated, PMCprimo uses the inputs from input byte 1. If a network subscriber is not stated, the inputs on the PMCPrimo are used.

Example:
State of input 2, Byte 2:

PMCprimo user	Comment
`0.1>  MA500;WI2:2-;MA0<CR>`	
`0.1M`	The motor travels to the absolute position of 500, waits until the state of input 2 Byte 2 is logic zero and then travels to absolute Pos. 0.
`0.1M`	
`0.1>`	

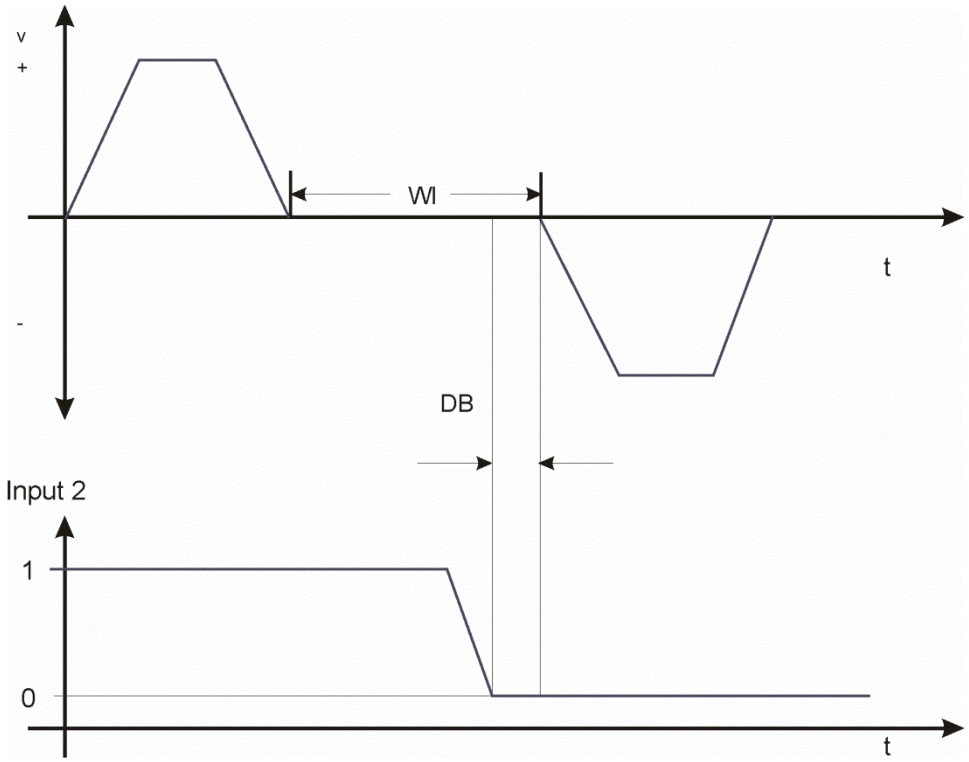


Figure 25: Wait for input line in group g

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.7 Wait loops

WA±nn Wait for absolute position

(**W**ait for **a**bsolute position)
Permitted values for n: ±2,000,000,000

This command can be used to program polling of an absolute position. PMCprimo waits until the required absolute position has been reached before executing the next command. The absolute position is stated in increments.

Example:
Change in velocity when an absolute position is reached.

PMCprimo user		Comment
`0.1>`	`SV200;VC+;WA1500;SV100<CR>`	
`0.1V`		The motor travels to the absolute position 1500 and, when the absolute position +1500 has been exceeded, changes the velocity from 200 to 100 increments/second.
`0.1W`		
`0.1V`		
`0.1>`		

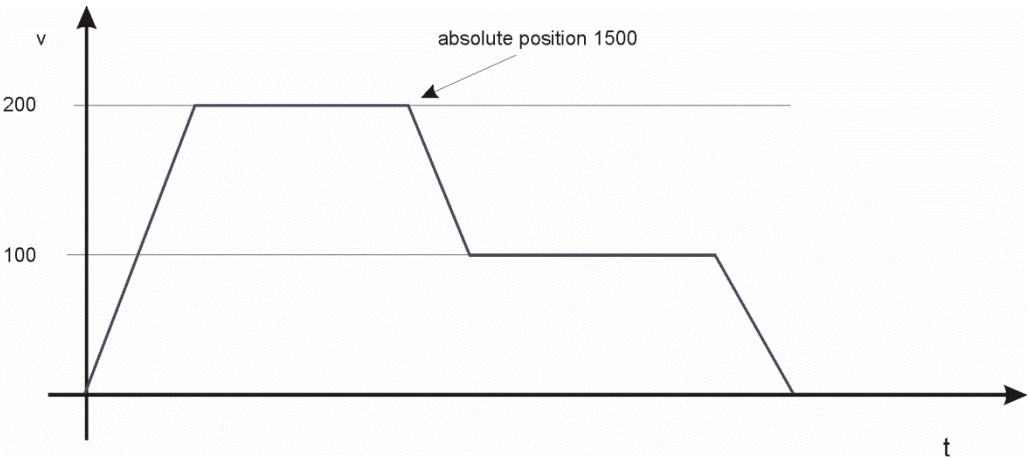


Figure 26: Wait for absolute position

3.7 Wait loops

WR±nn Relative position wait loop

(**W**ait for **r**elative position)

Permitted values for n: ±2,000,000,000

This command can be used to program polling of a relative position. PMCprimo waits until the required relative position has been reached before executing the next command. The relative position is stated in increments.

The wait for a relative position starts a WR position counter. When the content of this counter matches the value of "**nn**", the wait loop is ended.

Example:
Change in velocity when a relative position is reached:

PMCprimo user		Comment
0.1>	VC+;WR5000;SV1000;WR2000;ST<CR>	
0.1V		The motor travels 5000 increments in positive direction at constant velocity, changes the velocity and then executes the stop command after another 2000 increments.
0.1W		
0.1V		
0.1S		
0.1>		

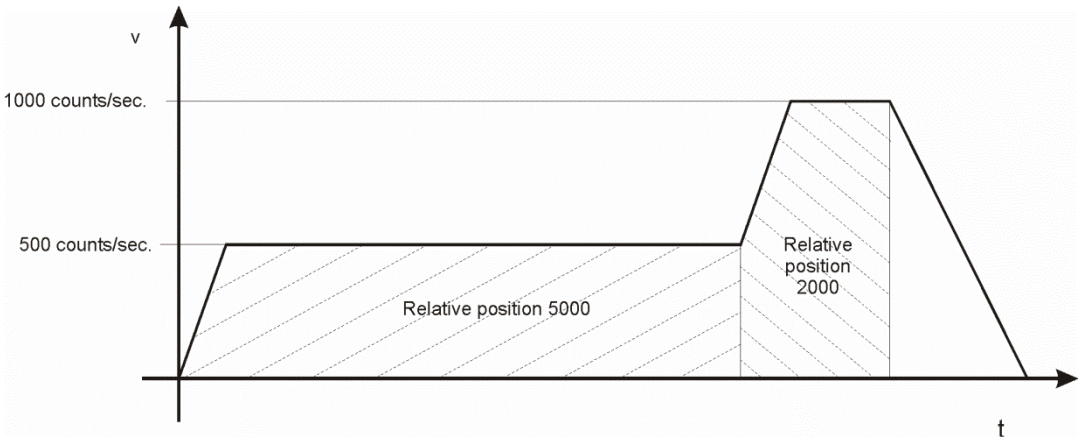


Figure 27: Wait for relative position

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.7 Wait loops

WF Wait for reference input

(**W**ait for re**f**erence input)

With this command, PMCprimo waits for a reference signal to be detected before executing the next command. This function is used in applications in which referencing must be changed following initialisation, for example.

Example:
Wait for reference input

PMCprimo user		Comment
0.1>	CH1;RW100001;IN+;WF;RW1<CR>	
0.1I		The motor initialises in positive direction
0.1W		and only corrects the position counter, not the motor position.
0.1>		After initialisation, the motor position will also be corrected at each reference signal.

WB Wait for bound position

(**W**ait for **b**ound position)

With this command, PMCprimo waits for the cycle limit to be exceeded and for this to be detected before executing the next command (see also SB command, Page 3-120).

Example:
Poll cycle limit

PMCprimo user		Comment
0.1>	XM;WB;ST<CR>	Position mapping is active on the slave axis.
0.1X		The axis stops when the cycle limit is reached.
0.1W		
0.1S		
0.1>		

3.7 Wait loops

WSn Wait for motor status

(**W**ait for motor **s**tatus)

With this command, PMCprimo waits until the motor status "**n**" is detected before executing the next command (possible values for motor status can be found in Chapter 4.3 on Page 4-253).

Example:
Wait for motor status "**PC**"

PMCprimo user	Comment
`0.1> mr100000`	Set motor status to positioning command
`0.1>`	
`0.1M ws0;sv`	Wait for PC status and then output SV value
`0.1M`	
`0.1>`	PC status has been reached
`0.1 SV=5000`	Output SV

WC±nn Wait for bound overflow count

(**W**ait for bound overflow **c**ount)

Permitted values for nn: ±2,000,000,000

With this command, PMCprimo waits for the overflow counter to assume the value "**±nn**" before executing the next command (see also SB command, Page 3-120). The overflow counter counts the number of times the cycle limit is exceeded in a positive or negative count direction, depending on the direction of the axis.

Example:
Wait for bound overflow count

PMCprimo user	Comment
`0.1> VC+;WC10;ST<CR>`	
`0.1V`	The axis travels in pos. direction until the overflow counter has reached the value 10,
`0.1W`	then executes the
`0.1S`	stop command.
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.7 Wait loops

WE End wait state

(**E**nd **w**ait state)

With this command, PMCprimo ends the current wait loop at the current axis and executes the next command immediately. The “**GE**” command (Page 3-71) can be used to cancel all active wait loops.

Example:
End wait state

PMCprimo user		Comment
`0.1>`	`ES P_WE`	Program triggers “**WE**” command.
`0.1>`	`CH0.1; WE`	
`0.1>`	`NS`	
`0.1>`	`DI1:1-,P_WE<CR>`	End wait loop with falling edge at input 1.

3.8 Monitoring functions

Setting error limits and defining the reaction of PMCprimo in the event of an error.

PMCprimo constantly monitors and checks its functions for errors. Some errors are critical because they stop PMCprimo controlling the drive correctly or they are errors such as the failure of a rotary encoder track. Other error sources may be categorised as less serious.

Critical errors are categorised as "Motor-OFF" errors. If a "Motor-OFF" error occurs, PMCprimo switches off the motors' controller enables. This is the safest method when a "Motor-OFF" error occurs. A corresponding error message is displayed on PTerm. The following errors are assessed as Motor-OFF errors:

- Position error limit exceeded
- Rotary encoder signals missing
- Limit switch reached

The following errors can be declared as Motor-OFF errors if selected as such via the EW command:

- Reference signal time monitoring
- Reference error too great
- Reference signal before existing reference error has been compensated

If these errors are not declared as Motor-OFF errors, all that will happen when they occur is that a corresponding error message will be output to the serial interface.

SEnn Set maximum position error

(**S**et maximum position **e**rror)

Permitted values for nn: 1 to 2,000,000,000
Factory setting: 1000

This command is used to set a maximum permitted positional deviation during the movement. If the set error limit is exceeded, PMCprimo brakes the axis to standstill and switches off the controller enable (power to the motor is removed). The position error limit is stated in increments.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.8 Monitoring functions

TOnn Set timeout

(Set **t**ime**o**ut)

Permitted values for nn: 1 to 2,000,000,000
Factory setting: 500

This command is used to set monitoring for the rotary encoder signals. The parameter can be interpreted differently, depending on the pre-setting through Bit 6 of the "**CW**" command (Page 3-54):

Bit 6 of "CW" is 0:
Once a movement has started, PMCprimo must receive a rotary encoder signal within the set monitoring time. If PMCprimo does not receive a rotary encoder signal before the monitoring time has elapsed, PMCprimo switches off the controller enable (power to the motor is removed). The monitoring time for rotary encoder signals is entered in milliseconds.

Bit 6 of "CW" is 1:
Once a movement has started, PMCprimo must receive a rotary encoder signal within the set path (setpoint specification). If PMCprimo does not receive a rotary encoder signal within the set path, PMCprimo switches off the controller enable (power to the motor is removed). The path for rotary encoder signals is entered in increments.

3.8 Monitoring functions

LHnn Set position limit high

(Set position **l**imit **h**igh)

Permitted values for nn: ±2,000,000,000
Factory setting: + 2,000,000,000

This command is used to set a software end limit in the positive direction. PMCprimo only starts a positioning movement if the target position is within the software end limit. If the software end limit is reached, PMCprimo switches off the controller enable (power to the motor is removed). The software end limit is stated in increments.

LLnn Set position limit low

(Set position **l**imit **l**ow)

Permitted values for nn: : ±2,000,000,000
Factory setting: – 2,000,000,000

This command is used to set a software end limit in the negative direction. PMCprimo only starts a positioning movement if the target position is within the software end limit. If the software end limit is reached, PMCprimo switches off the controller enable (power to the motor is removed). The software end limit is stated in increments.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.8 Monitoring functions

RTnn Set reference timeout

(Set **r**eference **t**imeout)

Permitted values for nn: 0 to 255
Factory setting: 0

This command can be used to activate monitoring for reference signals. If "**n≠0**", "**n**" reference signals may be lost before PMCprimo issues the error message "Monitoring time for the reference signal has elapsed!". If Bit 0 of the EW command (Page 3-110) is set to 1, PMCprimo switches off the controller enable (power to the motor is removed). This can also be used to monitor initialisation (IN; IB Page 3-39, 3-41). Monitoring only applies to signals that are not filtered out ("**FR**" command, Page 3-125).

If Bit 4 of "**FW**" is set and "**FR**" does not equal 0, the reference signal will be monitored as follows:

If the current position moves competely away from the range RF+FR to RF-FR, the monitoring counter is increased; at a counter status of "n", the following error message is issued: "Monitoring time for the reference signal has elapsed!".

Example:

PMCprimo user		Comment
0.1>	RT3<CR>	If 3 reference signals in succession are missing,
0.1>		PMCprimo detects this as an error.
0.1>	RT0<CR>	Absence of reference signals is not monitored.

3.8 Monitoring functions

EWbb Set error options word

(Set **e**rror options **w**ord)

Permitted values for bb: 0 to 1111 1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed. With this command the user can determine the behaviour of PMCprimo in the event of an error (Bit 0 is on the right with input/output and on the left with Bit 7).

Bit 0: Reaction of PMCprimo when the monitoring time for reference inputs is exceeded (see also RT command, Page 3-109).
0: In the event of an error, PMCprimo issues a signal.
1: In the event of an error, PMCprimo issues a signal and switches off the control enable.

Bit 1: Reaction of PMCprimo when a reference input is detected outside the permitted range (see also LR command, Page 3-125).
0: In the event of an error, PMCprimo issues a signal.
1: In the event of an error, PMCprimo issues a signal and switches off the control enable.

Bit 2: Reaction of PMCprimo when a reference signal is detected before an existing reference error has been adjusted.
0: In the event of an error, PMCprimo issues a signal.
1: In the event of an error, PMCprimo issues a signal and switches off the control enable.

Bit 3: Reaction of PMCprimo when the maximum permitted limit values at the analogue input are exceeded.
0: In the event of an error, PMCprimo issues a signal.
1: In the event of an error, PMCprimo issues a signal and switches off the control enable.

Bit 4: Suppress the error message “Reference input missing” (see also RT command, Page 3-109) if Bit 0 is not set to 1.
0: The error message is displayed.
1: The error message is not displayed if Bit 0 is set to 0.

Bit 5: Suppress the error message “Reference error limit exceeded” (see also LR command, Page 3-125) if Bit 1 is not set to 1.
0: The error message is displayed.
1: The error message is not displayed if Bit 0 is set to 0.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Bit 6: Suppress the error message “Reference signal detected before the existing reference was completely adjusted” if Bit 2 is not set to 1.

0: The error message is displayed.

1: The error message is not displayed if Bit 0 is set to 0.

Bit 7: Suppress the error message “Analogue input limit exceeded” (see also AH and AL command, Page 3-210) if Bit 3 is not set to 1.

0: The error message is displayed.

1: The error message is not displayed if Bit 0 is set to 0.

3.9 Control parameters

3.9 Control parameters

The drive is operated in position control during all positioning operations. Any discrepancies between the actual position and the constant or consecutive demand position will be resolved by the position controller.

Control algorithm:
PMCprimo contains a PID position controller with a reverse, velocity and acceleration pre-control factor:

$$V_{Set} = KP\ e_i + KI\ \Sigma e_i + KD\ (e_i - e_{i-1}) + KV\ (p_i - p_{i-1}) + KF(d_i - d_{i-1}) + KA[(d_i - d_{i-1}) - (d_{i-1} - d_{i-2})]$$

KP = Proportional gain
KI = Integral gain
KD = Differential gain
KV = Setpoint gain (velocity reverse amplification)
KF = Actual value gain (velocity pre-control gain)
KA = Acceleration gain (acceleration pre-control gain)
e_i = Lag distance (= demand position – actual position)
d_i = Demand position
p_i = Actual position

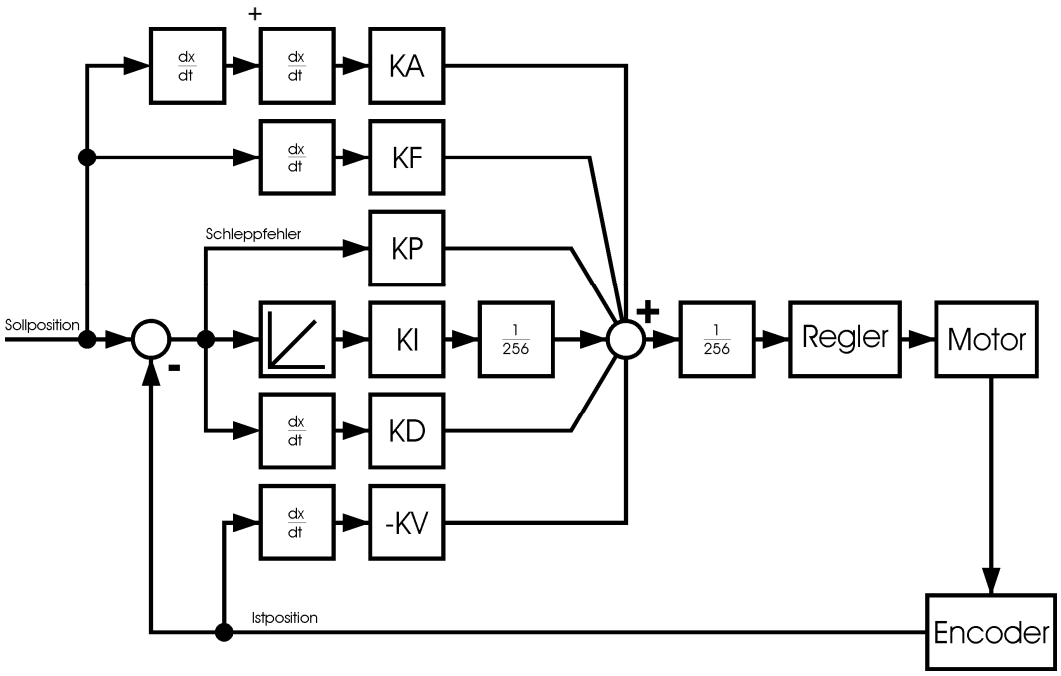
The dynamic behaviour of the drive depends on these constant gains and on the mechanical behaviour of the driven machine. These gains must be set in order to achieve optimum control behaviour.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.9 Control parameters

For a proportional controller, the following relationship applies between lag distance and setpoint output:

V_{Set} = Lag Distance * KP/256 * 10/2048



Regler - Controller

Figure 28: Relationship between lag distance and setpoint output

KPnn Set proportional gain constant

(Set **p**roportional gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 10

This command is used to set the proportional gain of the control algorithm. Setting a high value for the KP gain enables a short reaction time and precise position control. The KP gain should therefore be set as high as possible, without causing an overshoot.

3.9 Control parameters

KInn Set integral gain constant

(Set **i**ntegral gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 0

This command is used to set the integral gain of the control algorithm. The integral gain is useful for compensating a constant positional deviation due to continuous load and vibration or during velocity control. However, the integral gain also causes the axis to travel beyond the target position during positioning, because the positional deviation created during movement can accumulate. This problem is also known as “wind-up”.

KDnn Set differential gain constant

(Set **d**ifferential gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 0

This command is used to set the differential gain of the control algorithm. The differential gain is useful when there is a huge variation in position errors, e.g. with step-by-step position changes. In a well tuned control loop, the KD gain is of only minor importance.

KVnn Set velocity feedback gain constant

(Set **v**elocity feedback gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 0

This command is used to set the actual value gain of the control algorithm. The actual value gain scales the motor velocity resulting from the measured position. Use of the KV gain can be compared with the effect of a tachometer encoder. The KV gain has a damping effect, enabling a higher KP gain and thereby improving the reaction speed of the axis.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.9 Control parameters

KFnn Set velocity feed-forward gain constant

(Set velocity feed-**f**orward gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 0

This command is used to set the setpoint gain of the control algorithm. The setpoint gain influences the difference between the demand velocity and the actual measured velocity. With a pure proportional controller, a constant position error results when travelling at constant velocity. By specifying a KF gain it is possible to reduce the position error to zero or even into the negative range. The KF gain is added to the required velocity, so that the actual measured velocity equals the calculated demand velocity.

With standard settings, a value of 3750 is recommended for the PMCtendoDD4 and a value of 4465 for the PMCProtego / PMCtendoDD5.

KAnn Set acceleration feed-forward gain constant

(Set acceleration feed-forward gain constant)

Permitted values for nn: 0 to 65535
Factory setting: 0

This command is used to set the acceleration gain of the control algorithm. This gain uses the set acceleration in contrast to the actual acceleration and is helpful when the position counter is set or a velocity profile is run. The effect of "**KA**" is to provide a set value output component in proportion to the necessary set acceleration.

3.10 Reference commands

Activation and setting of the reference behaviour.

This chapter describes the commands and options available to the user from the PMCprimo in terms of referencing. Reference commands cover the basic initialisation function following a power failure, like all positioning systems, but also include the option for cyclical referencing, which enables the user to adjust fluctuations and inaccuracies of the actual cycle length to the theoretical cycle length specification.

The position of the rotary encoder is saved as soon as a reference input is detected. This saved position is compared with an anticipated reference position (cycle limit: -SBn / 0 / +SBn; see SB command on Page 3-120 / RL; see RL command on Page 3-121). The resulting difference is defined as a reference error; the absolute position can be corrected by this difference if required.

PMCprimo supports 2 types of reference signal:

A: The zero track of the rotary encoder is connected to a fast reference input, which even detects pulses with a length of 60 ns. This reaction time is short enough to detect the zero track for high resolution rotary encoders at maximum speed (Definition > Zero track is reference input < with DZ command).

B: Digital inputs as reference inputs. Inputs 1 to 6 can be defined with the DR command as reference inputs. With these reference inputs, external switching elements with a minimum pulse length of 50 µs can be used for referencing.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

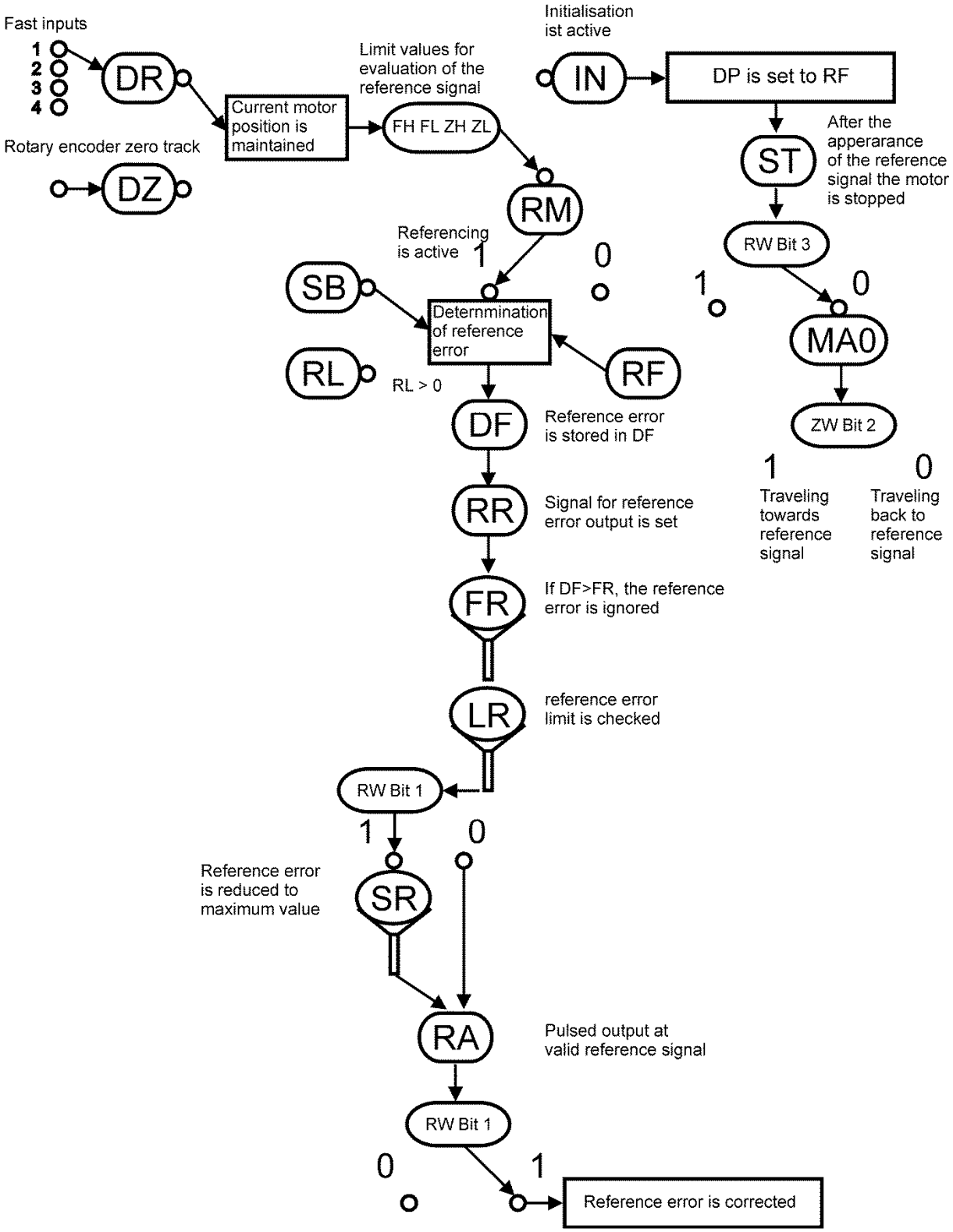


Figure 29: Interaction between the commands when a reference signal is detected

3.10 Reference commands

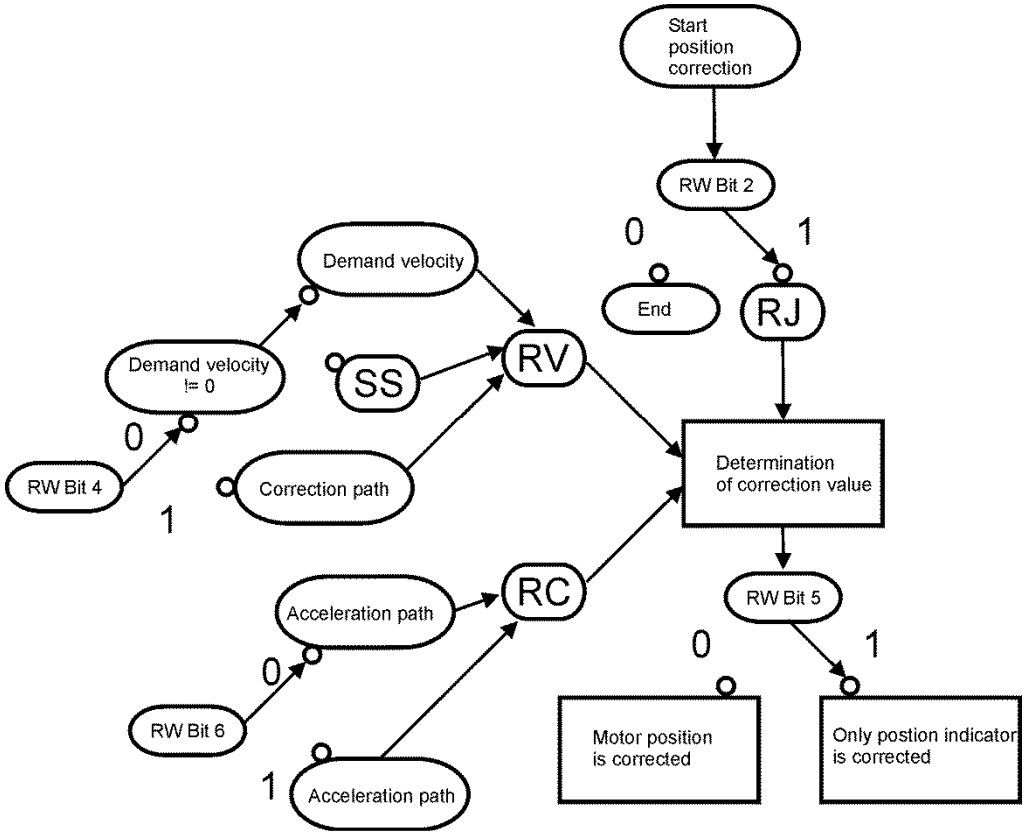


Figure 30: Interaction between the reference error correction commands

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

ZC[±nn] Zero position counters or set position

(**Z**ero position **c**ounters or set position)

Permitted values nn: ±2,000,000,000, or no details

This command sets the position counter to the stated value. If "**nn**" is not stated, the counter is set to the value zero.

Example 1:

PMCprimo user		Comment
`0.1>`	`MA-5000;ZC<CR>`	The motor travels to the absolute position -5000 and then the
`0.1>`		position counter is set to zero.

Example 2:

PMCprimo user		Comment
`0.1>`	`ZC1000<CR>`	The position counter is set to the value 1000.
`0.1>`		

3.10 Reference commands

SBnn Set position overflow bound

(**S**et position overflow **b**ound)

Permitted values for nn: 1 to 2,000,000,000
Factory setting: 4,194,304:

This command defines a cycle length as the limit for the absolute position, i.e. if the motor position exceeds the set cycle length, the value “**nn**” is subtracted from the absolute value of the position counter.

The limit of the absolute position is usually set to be identical to the reference position. This makes it easier to continually monitor the reference signals and carry out the necessary corrections that result.

Each full cycle length (limit value exceeded) is counted in an overflow counter. The BC command (Page 3-121) can be used to delete or set this counter; the WC command (Page 3-104) can be used to wait for a certain counter status. (Application example: parts counter).

Example:

PMCprimo user		Comment
`0.1>`	`SB1000<CR>`	Cycle length of 1000 increments set.
`0.1>`		

A typical application for using cycle lengths is a cyclical or rotating movement, in which only the position within one rotation or cycle is important. Positional details outside of the selected cycle length are possible at any time on travel commands; however, the current position at the end of the movement can never be greater than the cycle length (set value is subtracted from the position counter). The overflow counter BC contains the number of completed cycles.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

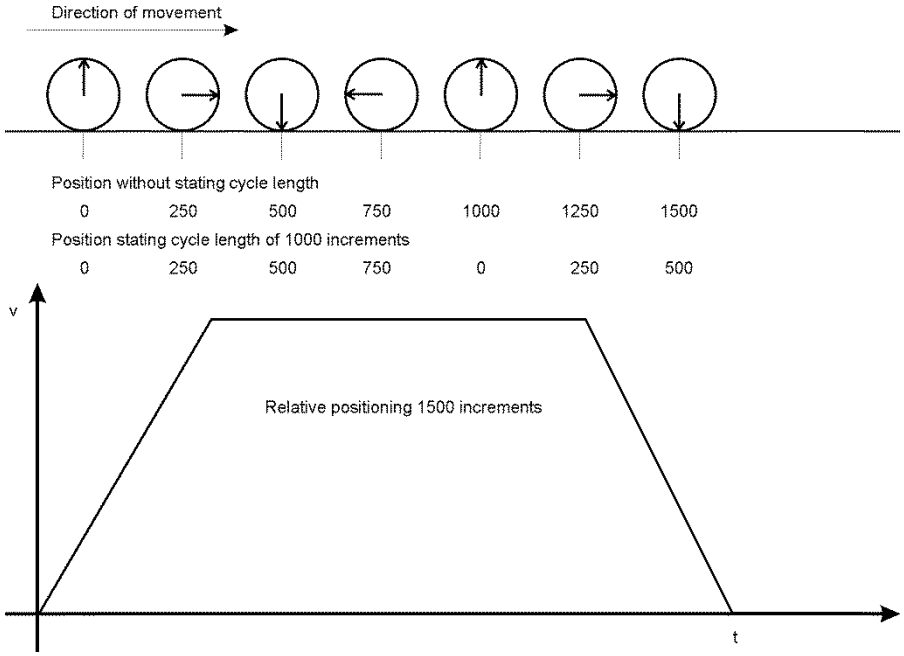


Figure 31: Cycle length

BC±nn Set/display position overflow counter

(Set/display position overflow **c**ounter)

Permitted values for nn: ±2,147,483,647

This command sets the overflow counter to the value "**nn**". The overflow counter counts the number of times the cycle lengths are exceeded. The counter value is incremented when the cycle limit is exceeded in a positive direction and decremented when it is exceeded in a negative direction.

RLnn Set reference repeat length

(Set **r**eference repeat **l**ength)

Permitted values for n: 0 to 2,000,000,000
Factory setting: 0

This command can be used to set a reference cycle length. The reference cycle length indicates the position at which the PMCprimo expects a reference signal. If a reference cycle length is not stated, PMCprimo will expect a reference signal at the cycle limit (set with SB command, Page 3-120).
Application example:
The machine's cycle length is "**(SB)10000**" and the reference cycle length is set with "**RL2500**". As a result, PMCprimo corrects a reference error 4 times within the machine's cycle length.

3.10 Reference commands

RMn Set continuous reference mode on/off

(Set continuous **r**eference **m**ode on/off)

Permitted values for n: 0 and 1
Factory setting: 0

With this command, when "**n=1**", the following are registered in each cycle (defined with SB command): the zero track as reference input (defined with DZ command, Page 3-128), all defined reference inputs (defined with DR command, Page 3-135) and all snapshots of the encoder position of the inputs defined using the PS command (Page 3-163).

RWbb Set reference options word

(Set **r**eference options **w**ord)

Permitted values for bb: 0 to 1111.1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

This command is used to select a function and preset the reference behaviour of PMCprimo (Bit 0 is on the right with input/output and on the left with Bit 7):

Bit 0: This bit activates cyclical referencing, if cyclical referencing has been activated using the RM command.

0: Cyclical referencing inactive.

1: Cyclical referencing active.

Bit 1: This bit determines the reaction of PMCprimo if the reference error to be corrected is greater than the maximum permitted value for reference error correction (SR command, Page 3-124).

0: PMCprimo ignores a reference error completely. Only reference signals within the "**SR window**" are evaluated and corrected.

1: PMCprimo executes a reference error correction with the maximum permitted value and issues the error message "**RL**". The user can define any further reaction, e.g. Motor OFF, in an error program (ME command, Page 3-71).

Bit 2: This bit activates a permitted position for correcting a reference error (setting with RJ command, Page 3-128).

0: No position is defined for correcting a reference error. The correction is made as soon as a reference input is detected.

1: Position for correcting a reference error is active.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

Bit 3: This bit prevents any movement back to the reference signal during initialisation (IN command, Page 3-39).

0: Movement back to the reference signal during initialisation is executed.

1: Movement back to the reference signal during initialisation is not executed.

Bit 4: This bit determines the functionality of the "**RV**" command (Page 3-126):

0: The velocity for the reference error correction is specified via "**RV**".

1: The path for the reference error correction is specified via "**RV**". In this case the path is the actual path, without correction. With this function, the reference error is always distributed on a defined, optimum path. With values greater than 200, a warning is issued when Bit 4 is switched from 1 to 0, because in this case, RV must be <200. "**RV**" is automatically set to 100.
If this bit is set, then Bit 6 of "**RW**" must also be set. If this is not the case, a warning is issued and Bit 6 is set automatically.

Bit 5: This bit determines whether a reference error correction is executed in the position indicator alone or also on the motor. This bit has no effect on a slave axis with active position mapping, as in this case, every reference error is corrected on the slave axis.

0: Motor position and position indicator are corrected.

1: Only the position indicator is corrected.

Bit 6: This bit determines the functionality of the "**RC**" command (Page 3-127):

0: "**RC**" is used to set the acceleration for the reference alignment run.

1: "**RC**" is used to specify an acceleration path that the axis would cover without an alignment run. If the value is 0, the alignment is made with a jump to "**RV**". With values between 1 and 1000, a warning is issued when Bit 6 is switched from 1 to 0, because in this case, RC must be >1000. "**RC**" is automatically set to 0.

Bit 7: Determines the correction of the position counter when a reference signal is detected.

0: PMCprimo searches for the shortest path for correcting the reference error. It will either be corrected at Position 0 or at the end of the cycle limit. The prefix of the reference error determines the direction of the alignment.

1: PMCprimo always corrects the reference error to the value 0. The prefix of the reference error is always negative.

3.10 Reference commands

SRnn Set maximum reference correction

(**S**et maximum **r**eference correction)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

When "**n≠0**", this command limits the value of the reference error that is corrected on the motor to the value of "**nn**". This command can be used to eliminate incorrect reference signals lying far away from the expected position, or to enable cyclical referencing, even though the reference signals do not match the set cycle length.

If PMCprimo detects a reference signal, the reference error is calculated as the difference between the zero point defined with the reference signal and the zero position, or the SB position.

If the reference error is within "**SR**" and Bit 0 of "**RW**" (Page 3-122) is 1, the reference error is offset on the motor.

If the reference error is outside the limit value set with "**SR**", the reaction of PMCprimo will depend on Bit 1 of the RW command and value "**n**" of the FR command (Page 3-125):

If Bit 1 of "**RW**" is set to 0, the reference error will not be offset and the error message "Reference error limit exceeded!" will not be triggered. The reference error will be ignored completely.

If Bit 1 of "**RW**" is set to 1, the value of "**SR**" will be offset as the maximum error at the motor. The error message "Reference error reduced to the maximum value!" will be triggered!"

If the value of "**LR≠0**", the error message "Reference error limit exceeded!" will be triggered if the reference error is greater than the value of "**LR**". Bit 1 of the EW command (Page 3-110) can be used to determine whether the motor is to be shut down in addition to this error message.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

FRnn Set filter on reference error

(Set **f**ilter on **r**eference error)

Permitted values for n: 0 to 65535
Factory setting: 0

This command can be used to set a filter for a reference error. If "**n=0**", no filter is active. When values for "**n≠0**", PMCprimo totally ignores reference errors that are greater than "**FR**". The FR command is used to filter out reference signals that are outside the expected permitted range. The FR command is independent of "**SR**" (Page 3-124). "**SR**" limits the correction value that is corrected in the event of a reference error on the motor.

LRnn Set reference error limit

(Set **r**eference error **l**imit)

Permitted values for n: 0 to 65535
Factory setting: 0

This command can be used to set a reference error limit in increments. If PMCprimo detects a reference error that is greater than "**LR**", the error message "Reference error limit exceeded" is triggered. "**LR**" is independent of "**SR**" (Page 3-124) and "**FR**". Bit 1 of the EW command (Page 3-110) can be used to determine whether the motor is to be shut down in addition to this error message.

"**LR**" is monitored before "SR". This means that a message appears if "**LR**" is greater than "**SR**".

The message "Reference error reduced to the maximum value!" is not triggered if the value n≠0.

RF±nn Set reference offset

(Set **r**e**f**erence offset)

Permitted values for n: ±2,000,000,000
Factory setting: 0

This command can be used to set a reference offset. If "**n=0**", PMCprimo regards the position of the reference signal as the absolute position 0. If values for "**n≠0**", PMCprimo regards the position of the reference signal as the absolute position "**n**".

3.10 Reference commands

RVn Set reference correction velocity

(Set **r**eference correction **v**elocity)

Permitted values for n: 0 to 200%
Factory setting: 0

To correct a reference error, a temporary change in velocity is required while a movement is executed. PMCprimo can use the RV factor to limit the maximum velocity while a reference error is corrected. If n=0 is stated, the reference error correction is processed as a step function. The velocity "**RV**" is stated as a % of the current demand velocity. If a reference error is detected when the axis is stationary, the error will be offset with the velocity of "**SS**" (Page 3-48). The following relationship applies for values "**n≠0**":

Change in velocity = Current velocity x RV/100

If PMCprimo detects a reference signal before the existing reference error has been offset, the following error message is generated:

"Could not offset the reference error in one cycle".

This error can occur if the change in velocity is too small and the cycle lengths are very short. If Bit 2 of the EW command (Page 3-110) is set, PMCprimo will shut down the motor in the event of an error.

The accelerator for processing a reference error (see Fig. 33) can be set using the "RC" parameter.

.

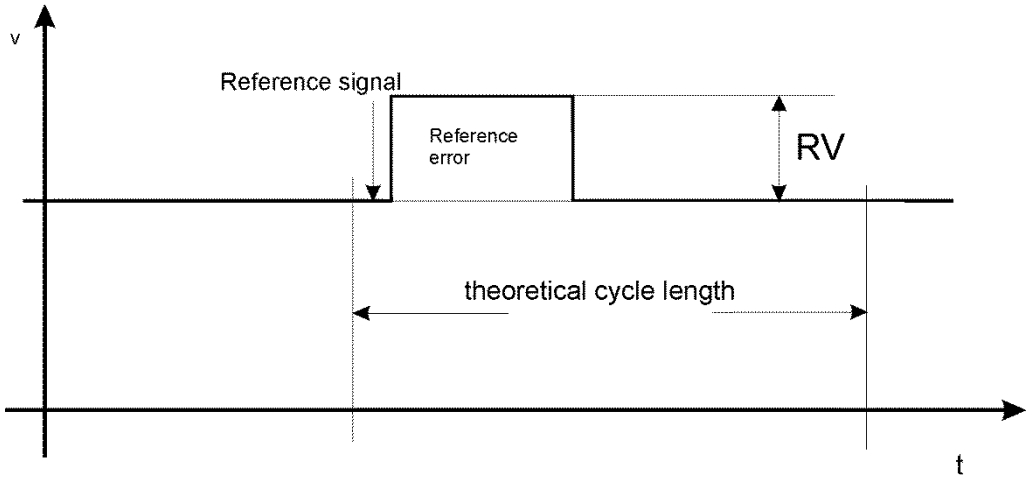


Figure 32: Reference error correction with RV as velocity gain

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

If Bit 4 of the “**RW**” command is set (Page 3-122), the path for the reference error correction is specified via “**RV**”. In this case the path is the actual path, without correction. With this function, the reference error is always distributed on a defined, optimum path. With values greater than 200, a warning is issued when Bit 4 is switched from 1 to 0, because in this case, RV must be <200. “**RV**” is automatically set to 100.

If Bit 4 of “**RW**” is set, then Bit 6 of “**RW**” must also be set. If this is not the case, a warning is issued and Bit 6 is set automatically.

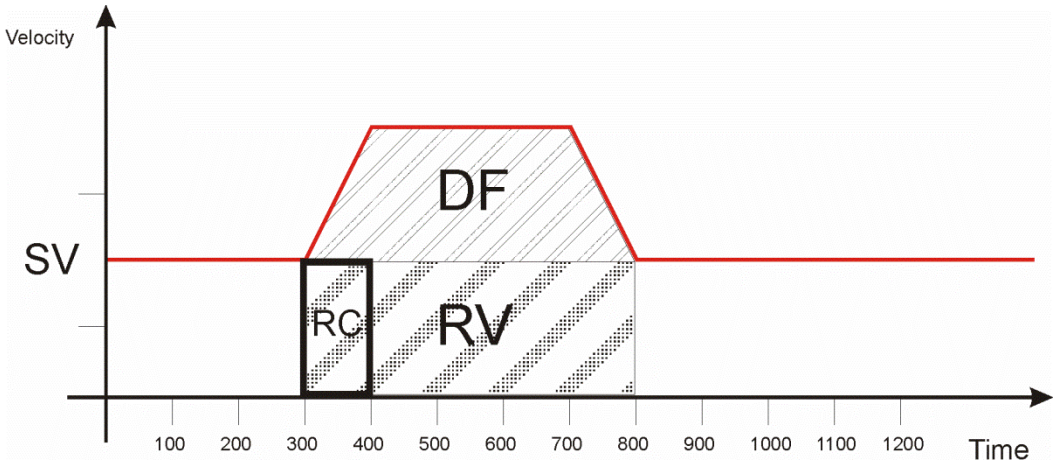


Figure 33: Reference error correction with RV as path

The overall path “**RV**” and the acceleration path “**RC**” are specified. The necessary acceleration and velocity to compensate the reference error “**DF**” are calculated from these automatically.

RCn Set reference error adjustment acceleration

(Set **r**eference error adjustment a**c**celeration)

Permitted values for n: 1 to 2,000,000,000
Factory setting: 0

Adjusting a reference error results in a new demand position for the slave axis. PMCprimo can use the “**RC**” parameter to determine the acceleration ramp for achieving the adjustment velocity. The “**RC**” parameter is stated in increments / second^2. The “**RC**” parameter should be set at the slave axis. If “**n**” is not stated, the acceleration is a step.

If Bit 6 of the “**RW**” command is set (Page 3-122), “**RC**” is used to specify an acceleration path, which the axis would cover without an alignment run. If the value n is 0, the alignment is made with a step to “**RV**”. With values between 1 and 1000, a warning is issued when Bit 6 is switched from 1 to 0, because in this case, RC must be >1000. “**RC**” is automatically set to 0.

3.10 Reference commands

RJnn Set deferred reference adjustment position

(Set deferred **r**eference ad**j**ustment position)

Permitted values for n: 0 to 2,000,000,000
Factory setting: 0

With this command the user can define the position within the cycle limits, at which PMCprimo may correct a stored reference error. If n=0, PMCprimo corrects the reference error immediately. The position that can be specified using this command must be activated with Bit 2 of the preset reference behaviour (RW command, Page 3-122).

If a value n-times greater than "**SB**" (Page 3-120) or "**RL**" (Page 3-121), if used, is specified for "**RJ**", execution of the reference error correction is displaced by n cycles.

A relative or absolute position for "**RJ**" can be selected with Bit 1 of the "**FW**" command (Page 3-128).

DZn Define zero marker input on/off

(**D**efine **z**ero marker input on/off)

Permitted values for n: 0 and 1
Factory setting: 0

This command can be used to define the zero track of the rotary encoder as the reference input. If n=1 is stated, PMCprimo evaluates the zero track of the rotary encoder as a reference input.

RNnn Set phase advance for reference signal

(Set phase adva**n**ce for **r**eference signal)

Permitted values for n: ±2,000,000,000
Factory setting: 0

This command can be used to provide an axis reference signal with a velocity-dependent phase displacement.

The calculation is made in accordance with the following formula (calculation as for the "**PA**" and "**BA**" commands (Page 3-201)):

$$\text{Phase displacement reference signal} = \frac{\text{Demand velocity}}{256} \times \frac{RN}{256}$$

The average demand velocity is used for the calculation. The interval "**VT**" (Page 3-201) is used in this case, as set with the command.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

FWbb Set reference filter options word

(Set reference **f**ilter options **w**ord)

Permitted values for bb: 0 to 1111.1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

This command is used to preset the options for a PMCprimo reference filter (Bit 0 is on the right with input/output and on the left with Bit 7):

Bit 0: This bit activates the parameters “**ZH**”, “**ZL**”, “**FH**”, “**FL**”.
0: Parameters “**ZH**”, “**ZL**”, “**FH**”, “**FL**” inactive.
1: Parameters “**ZH**”, “**ZL**”, “**FH**”, “**FL**” active.

Bit 1: This bit determines the reference position of “**RJ**” (Page 3-128) in PMCprimo.
0: PMCprimo uses the relative position “**RJ**” for the reference signal.
1: PMCprimo uses the position of the position counter for “**RJ**”.

Bit 2: This bit defines the output of the measured reference error via “**DF**”.
0: “**DF**” displays the measured reference error.
1: “**DF**” does not display the filtered reference error.

Bit 3: This bit decides the functionality of the debounce time “**RH**” for the reference signal.
0: Value is used as the time for calculating the debounce time.
1: Value is used as the distance for calculating the debounce time. A further reference signal is ignored if the distance between the last signal and the current signal is less than the value “**RH**” (Page 3-134).

Bit 4: This bit decides how the reference signal is monitored.
0: Monitoring with “**RT**” for round axis.
1: Monitoring with “**RT**” for linear axis.
If the current position moves away from the range RF+FR to RF-FR, the monitoring counter is increased and, if necessary, an error message is triggered.

Bit 5: This bit decides on a variable distance for the reference error correction
0: Fixed distance for reference error correction
1: Variable distance for reference error correction and fixed distance for correction-free zone

3.10 Reference commands

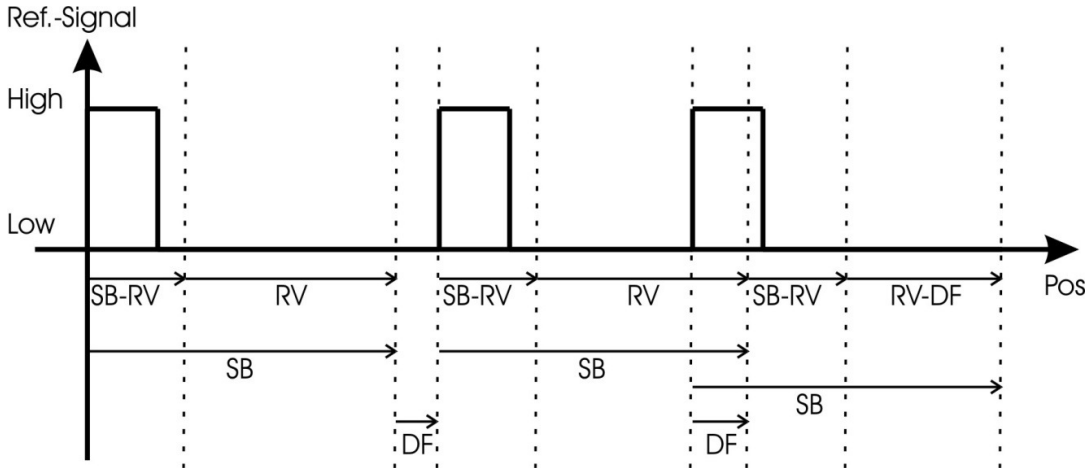


Figure 34: Variable distance for reference error correction

SB: Cycle length
SB-RV: Fixed distance for correction-free zone
RV: Maximum distance for reference error correction
DF: Reference error
RV-DF: Variable distance for reference error correction

Bit 6: This bit can be used to select whether the debounce time or debounce distance RH is to affect ZY (bit not set) or ZX (bit set).

0: RH only affects ZY, if reference filter is active.
1: RH only affects ZX, if reference filter is active.

Bit 7: Special filter function for referencing:
The debounce distance RH (FW Bit 3 must be set) is effective from the falling edge of the reference signal. The reference signal is only detected as such once RH has elapsed. If RH is already active, RH is restarted by another falling edge.
To ensure that the falling edge is detected, Bit 0 of FW must be set and the signal length "high" must be defined using ZL and ZH.

0: Special filter function is inactive.
1: Reference signal is not detected until the debounce distance RH has been travelled after a falling edge

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

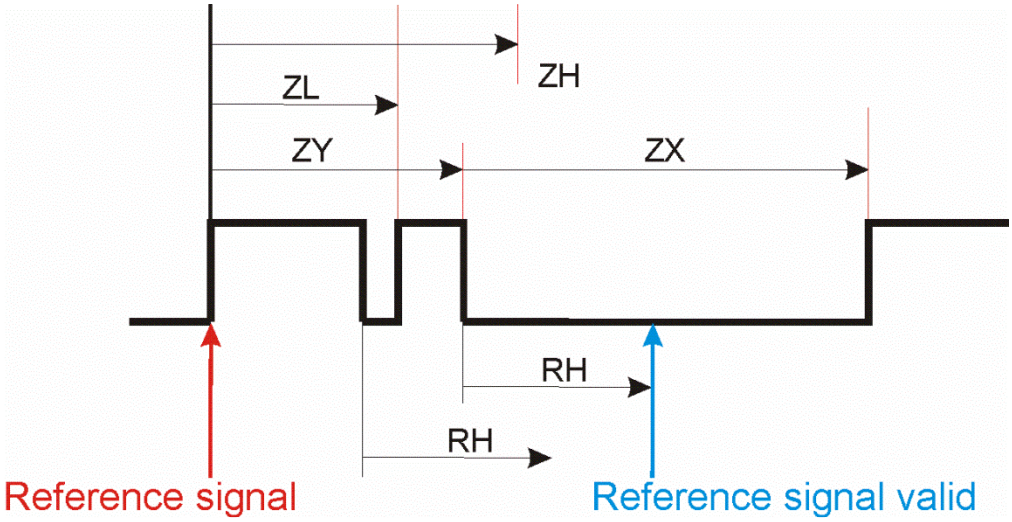


Figure 35: Special filter function for referencing

ZHnn **Set reference input true high limit**
(Set reference input true **h**igh limit)

ZLnn **Set reference input true low limit**
(Set reference input true **l**ow limit)

FHnn **Set reference input false high limit**
(Set re**f**erence input false **h**igh limit)

FLnn **Set reference input false low limit**
(Set re**f**erence input false **l**ow limit)

Permitted values for n: ±2,000,000,000
Factory setting: 0

These four parameters limit the travel length of a reference input defined using the DR command. They enable a situation in which PMCprimo only reacts to a reference input if the signal is detected within the specified limits. The reference input is only assessed as valid if it is false between the values "**FL**" and "**FH**" and then true between the values "**ZL**" and "**ZH**". The input is only assessed by PMCprimo as a valid reference input if it has switched from true to false. However, the reference error for initialisation and cyclical referencing is detected with a rising edge at the input signal (switching from false to true). The path for "**n**" is stated in increments.

The limit values for reference input true are not checked when "**n=0**" is stated for "**ZH**" and "**ZH**". The limit values for reference input false are not checked when "n=0" is stated for "**FH**" and "**FL**". If n=0 is stated on all four parameters, the limit values are not checked.

Summary:

3.10 Reference commands

A reference input is evaluated as such if

- ZL <= Travel length with reference input (DR) true <= ZH and
- FL <= Travel length with reference input (DR) false <= FH

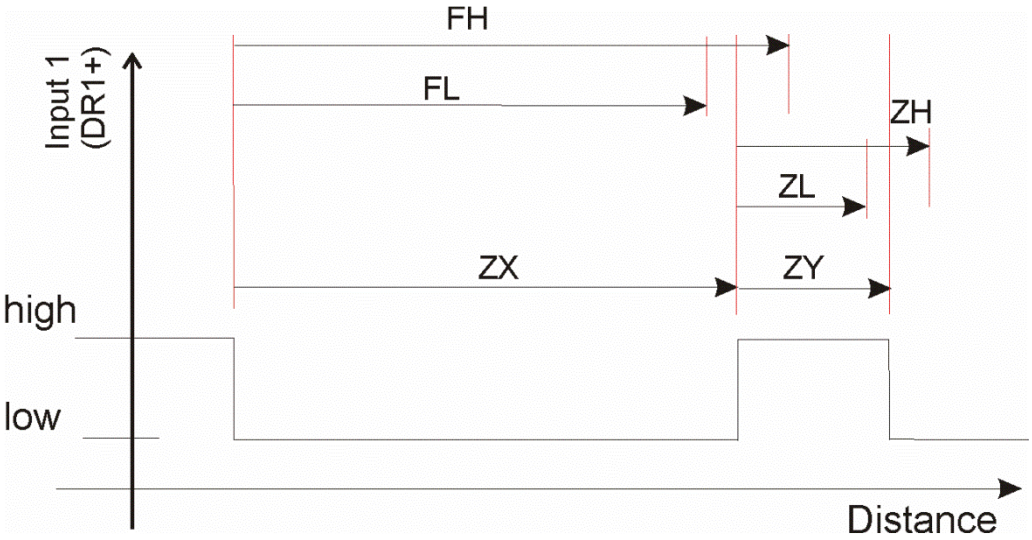


Figure 36: Limit values for evaluating the reference input

Reference input is evaluated as such when FL <= X <= FH and ZL <= Y <= ZH. This option is used in order to filter a valid reference signal from a spectrum of signals from the reference sensor. This filter property is especially suitable for printing and registering applications.

ZX Display length reference signal false

(Display length reference signal false)

With this command, the last measured length via which the reference signal had the status "**false**" is displayed. It is only displayed if Bit 0 of the FW command (Page 3-128) is set to 1. If Bit 0 of the FW command is set to 0, "**ZX**" is also set to zero automatically.

The edge to be detected is always changed internally, to enable the length of the reference signal to be measured. The reprogramming time is a minimum debounce time (RH function, Page 3-134). Signals that are shorter than this debounce time or 1 millisecond cannot be measured and therefore supply a false result.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

ZY Display length reference signal true

(Display length reference signal true)

With this command, the last measured length via which the reference signal had the status "**true**" is displayed. It is only displayed if Bit 0 of the FW command (Page 3-128) is set to 1. If Bit 0 of the FW command is set to 0, "**ZY**" is also set to zero automatically.

The edge to be detected is always changed internally, to enable the length of the reference signal to be measured. The reprogramming time is a minimum debounce time (RH function, Page 3-134). Signals that are shorter than this debounce time or 1 millisecond cannot be measured and therefore supply a false result.

RA[g:]n± Define reference accepted output

(Define **r**eference **a**ccepted output)

Permitted values for g: 1 to 8 (PMCprimo C)
Permitted values for n: 1 to 8

This command can be used to activate a pulsed output when a valid reference input signal is detected. If PMCprimo detects the signal from a reference input as valid, PMCprimo outputs a pulse of 1 ms at the output defined as "**n**" from Byte "**g**". The ± sign indicates the signal level of the output for the pulsed output. If + or - is not stated, a pulsed output can be cancelled. For example, this function can be used to recognise products as good, if the commands "**ZH**", "**ZL**", "**FH**" or "**FL**" (Page 3-131) are used.

A pulsed output also occurs during initialisation with "**IN**" and "**IB**" (Page 3-39 and 3-41).

3.10 Reference commands

RHnn Debounce time reference signal

(Set **r**eference **h**oldoff time)

Permitted values for n: 0 to 2,000,000,000
Factory setting: 0

This command can be used to set a debounce time for a reference signal. PMCprimo will not accept the next reference signal until this switch-off delay has elapsed.

If Bit 3 of the "**FW**" command is set to 1, the debounce time will no longer be used as a time but as a distance. In this case, "**RH**" is evaluated in increments. A further signal is ignored if the distance between the last signal and the current signal is less than the value "**RH**". As the distance always remains the same, bouncing signals that are based on the velocity can be filtered out better.
The debounce time is only effective when reacting to the rising edge (actual reference signal).
If Bit 0 of "**FW**" is set, Bit 6 of "**FW**" can be used to decide whether "**RH**" is effective for the "**ZX**" or "**ZY**" range.

OR[g:]n± Define reference output

(Define **r**eference **o**utput)

Permitted values for g: 1 to 8 (PMCprimo C)
Permitted values for n: 1 to 8

This command can be used to set an output to "true" for the time in which a reference error correction is executed. If + or - is not stated, a definition can be invalidated.

The output switches to "**1**" for 1 millisecond even if the reference error is 0.

JF[g:]n± Reference adjustment forwards output

(Reference ad**j**ustment **f**orwards output)

Permitted values for g: 1 to 8 (PMCprimo C)
Permitted values for n: 1 to 8

This command can be used to set an output for the time in which a reference error correction is executed in the travel direction. This means that the axis must be faster during the reference error correction so that the defined output is set to "true". If + or - is not stated, a definition can be invalidated.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.10 Reference commands

JB[g:]n± Reference adjustment against backwards output

(Reference adjustment backwards output)

Permitted values for g: 1 to 8 (***PMCprimo*** C)
Permitted values for n: 1 to 8

This command can be used to set an output for the time in which a reference error correction is executed against the travel direction. This means that the axis must be slower during the reference error correction so that the defined output is set to "true". If + or - is not stated, a definition can be invalidated.

RKn,p Reference position strange channel

(Reference position strange channel)

Permitted values for n: 1 to 60 Permitted values for p: ±2,000,000,000

With this command, the reference error correction of the current axis is executed at the position 'p' of the 'n' axis. The current axis waits until the axis 'n' has reached the position p; only then does it begin the reference error correction. “**RJ**” (Page 3-128) is no longer effective on the current axis. If the position 'p' is not stated, the definition is deleted.

3.10 Reference commands

DR[g:]n± Define reference input

(Define reference input)

Permitted values for g: 1
Permitted values for n: 1 to 6

This command can be used to define a digital input as a reference input. The definition only applies on the selected axis. The axis may also be virtual. All inputs from 1 to 6 can be defined as reference inputs. It is not necessary to state the byte address "**g**". If ± is not stated, an existing definition is invalidated.

Example 1:

PMCprimo user	Comment
`0.1>  DR1:1-<CR>`	This command line is used to define input 1 as the reference input of axis 1.
`0.1>`	PMCprimo evaluates the falling edge as the reference signal.

Example 2:

PMCprimo user	Comment
`0.1>  CH2;DR1:2+<CR>`	This command line is used to define input 1:2 as the reference input of axis 2. PMCprimo evaluates the rising edge as the reference signal.
`0.2>`	

Example 3:

PMCprimo user	Comment
`0.1>  LI1<CR>`	This command line is used to list the input definitions of Byte 1.
`# Inputs 0.1`	
`0.1:1 -R1`	
`0.1:2 +R2`	
`0.1:3`	
`0.1:4`	
`0.1:5`	
`0.1:6`	
`0.1:7`	
`0.1:8`	
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

Querying and linking of digital inputs and setting / resetting of digital outputs. PMCprimo has a number of freely available digital inputs and digital outputs, depending on the design.

PMCprimo C:	1 actual input group (with 6 inputs)
	7 virtual input groups
	1 actual output group (with 6 outputs)
	7 virtual output groups

The digital inputs and outputs are divided into groups of 8 (=Byte). Each address must be provided with a byte address. A colon is used to separate the byte address (g) and bit address (n).

Examples:

`SO1:2`	Set output 2 of Byte 1 of the PMCprimo C.
`SO1`	Set all unused outputs of Byte 1 of the PMCprimo C.
`CO1.2:5`	Reset output 5 of Byte 2 on network subscriber 1
`RI1`	Read the state of all inputs of Byte 1 of the PMCprimo C
`MI2.3:7`	Mask input 7 of Byte 3 on network subscriber 2
`EI1`	Enable all inputs of Byte 1 of the PMCprimo C.

Virtual inputs and outputs:
Virtual outputs are connected to virtual inputs through the software. With the exception of the “**DR**” and “**PS**” commands, all commands can be used for the virtual inputs/outputs. Even the debounce time “**DB**” affects the virtual inputs.
The virtual input banks 0.101 to 0.108 are mapped to the bus variables $B101 to $B108. Corresponding outputs, such as those found on classic virtual inputs, are not available. These virtual inputs are intended to receive input signals via a Modbus or Profibus.

Counters and timer functions are available for the digital outputs.

3.11 Digital input and output commands

SO[t.g:][n][X1,X2] **Set output line n in group g**

(**S**et **o**utput line n in group g)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8, or no details (PMCprimo C)
Permitted values for n:	1 to 8, or no details
Permitted values for X1:	1 to 8, or no details
Permitted values for X2:	0 to 256, or no details

PMCprimo uses this command to set the selected output to the logical signal state 1. If **"n"** is not stated, all non-defined outputs of Byte "**g**" are set to to the signal state 1. If "**g**" is not stated, PMCprimo uses output byte 1. If "**t**" is not stated, the PMCprimo is selected.

Parameters X1 and X2 can be used to set the outputs n to X1 in accordance with the bit pattern of X2.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

Example 1:

PMCprimo user	Comment

`0.1> so0.1:2,4,0x7<CR>`
`0.1> ro1`
`0.1>`
`# Outputs 0.1`
`8 7 6 5 4 3 2 1`
`0 0 0 0 1 1 1 0`
	Outputs 2 to 4 of output group 1 are assigned the value 1.

Example 2:

PMCprimo user		Comment
`0.1>`	`SO1:3<CR>`	This command line is used to set output 3 of Byte 1 to the logical signal state 1.
`0.1>`		

Example 3:

PMCprimo user		Comment
`0.1>`	`SO1<CR>`	This command line is used to set all the outputs of Byte 1 to the logical signal state 1.
`0.1>`		
`0.1>`	`SO<CR>`	This command line is used to set all the outputs, provided the outputs are not defined with a function.

3.11 Digital input and output commands

CO[t.g:][n] **Clear output line n in group g**

(**C**lear **o**utput line n in group g)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8, or no details (PMCprimo C)
Permitted values for n:	1 to 8, or no details

PMCprimo uses this command to set the selected output to the logical signal state 0. If "**n**" is not stated, all outputs of Byte "**g**" are set to to the signal state 0. If "**g**" is not stated, PMCprimo uses output byte 1. If "**n**" is not stated, all non-defined outputs are reset. Virtual outputs are connected to virtual inputs through the software.

Example 1:

PMCprimo user		Comment
`0.1>`	`CO0.2:7<CR>`	This command line is used to set output 7 of Byte 2 to the logical
`0.1>`		signal state 0.

Example 2:

PMCprimo user		Comment
`0.1>`	`CO1<CR>`	This command line is used to set all the outputs of Byte 1 to the logical
`0.1>`		signal state 0.
`0.1>`	`CO<CR>`	This command line is used to reset all the outputs, provided the outputs are not defined with a function.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

RI[t.g] Read input lines in group g

(**R**ead **i**nput lines in group g)

Permitted values for t: 0 to 150 (depending on the addresses of the network subscribers) or no details

Permitted values for g: 1 to 8 (PMCprimo C)

PMCprimo uses this command to show the state of the inputs of the byte defined as "**g**" on network subscriber "**t**".

In addition to showing the state of the logic signal, it also displays which input is masked (M), inhibited (B) or enabled (E). Virtual outputs are connected to virtual inputs through the software.

Example:

PMCprimo user	Comment
`0.1>  RI0.1<CR>`	The state of input bank 1
`1 2 3 4 5 6 7 8`	Inputs 1 to 8 of Byte 1
`0 1 1 0 0 1 0 0`	Logic signal state of inputs 1 to 8
`E E B M E B E E`	Enabled (E), masked (M), inhibited (B) inputs
`0.1>`	

RO[t.g] Read output line states in group g

(**r**ead **o**utput line states in group g)

Permitted values for t: 0 to 150 (depending on the addresses of the network subscribers) or no details

Permitted values for g: 1 to 8 (PMCprimo C)

PMCprimo uses this command to show the state of the outputs of Byte g. If g is not stated, the outputs of group 1 are displayed. Virtual outputs are connected to virtual inputs through the software.

Example:

PMCprimo user	Comment
`0.1>  RO1<CR>`	This command line is used to show the state of the outputs of Byte 1 of the PMCprimo C
`1 2 3 4 5 6 7 8`	
`0 1 1 0 0 1 0 0`	Logical signal state of outputs 1-8 is displayed.
`0.1>`	

3.11 Digital input and output commands

II[t.]g:n± If input true do command line

(**I**f **i**nput true do command line)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user can arrange PMCprimo so that a command is only executed if the state of the input defined as "**n**" of Byte "**g**" has the required logic state. If the input does not have the required state, PMCprimo continues processing in the following line of a program or processes the program section after the EL command in this line. Virtual outputs are connected to inputs through the software.

Example 1:

PMCprimo user	Comment
`0.1>  II2:2-;CO1:3<CR>`	This command line is used to poll the state of input 2 of Byte 2 for 0. If the input has the status 0, PMCprimo resets output 3 of Byte 1.
`0.1>`	

Example 2:

PMCprimo user	Comment
`0.1>  II1:2-;XS PRG3;EL;XS PRG21<CR>`	This command line is used to poll the state of input 2 Byte 1 for 0. If the input has the state 0, PMCprimo calls the program PRG3. If the state is 1, program PRG21 is called.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

IT If motor status true do command line

(**I**f mo**t**orstatus true do command line)

With this command, the user can arrange PMCprimo so that a command is only executed if the motor is in operation. If the motor is switched off, PMCprimo continues processing in the following line of a program or processes the program section after the EL command in this line.

Example:

PMCprimo user	Comment
`0.1>  IT;CO1:3<CR>`	Output 1:3 is set, as the motor is switched on.
`0.1>`	

IO[t.] g:n± If output true do command line

(**I**f **o**utput true do command line)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user can arrange PMCprimo so that a command is only executed if the state of the output defined as "**n**" of Byte "**g**" has the required logic state. If the output does not have the required state, PMCprimo continues processing in the following line of a program or processes the program section after the EL command in this line.

Example:

PMCprimo user	Comment
0.1> IO2:4+;XS PROG23<CR>	This command line is used to poll the state of output 2 of Byte 2 for 1. If the output is set, PMCprimo processes the program PROG23.

3.11 Digital input and output commands

MI[t.][g:n] **Mask input function**

(**M**ask function **i**nput)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user disables (masks) a defined input function with the byte address “**g**” of the input defined with “**n**” in PMCprimo. If “**n**” is not stated, all inputs of “**g**” are disabled. If “**n**”, “**g**” and “**t**” are not stated, all input functions are disabled. An input function can be masked, for example, to ensure that a program can only be started once. After this program has been called, this command is used to mask the input function, i.e. the command can only be repeated if the input function mask has been cancelled using the EI command. This method can also be used to mask input functions if they could disrupt the execution of other functions. (This command does not influence the commands “**II**” or “**WI**”!). If the state of a masked input changes, the respective input function will be executed after the input is enabled.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

Example:

PMCprimo user	Comment
0.1:	
0.1: ES PROG15	Entry of program PROG15
002E PR'PROG15 started'	Text output at program start
003E CH0.1;PC	Activate axis 1 position control
004E CH0.1;MR500	Execute relative positioning of 500 increments
005E NS	End of program entry
0.1: DI2:5+,PROG15	Determine input definition: Rising edge starts PROG15
0.1: RI2:5	Read input states
0.1:	
# Inputs 0.2	
8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0	All inputs FALSE
E E E E E E E E	Enable all input definitions
0.1: MI2:5	Mask input 2:5
0.1:	
	In the mean time input 2:5 has changed its state, PROG15 was not executed, as input masked.
0.1:	
0.1: RI2	Read input states
0.1:	
# Inputs 0.2	
8 7 6 5 4 3 2 1	
0 0 0 1 0 0 0 0	Input 2:5 TRUE
E E E M E E E E	Input 2:5 masked
0.1: EI2:5	Enable input 2:5
0.1:	PROG15 is started as input state at EI does not equal that at MI
PROG15 started	Text output
0.1>	Activate position control
0.1M	Execute relative positioning
0.1>	

3.11 Digital input and output commands

BI[t.][g:n] **Inhibit function input**

(Inhi**b**it function **i**nput)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user inhibits a defined input function (or an input group) with the byte address "**g**" of the input defined with "**n**" in PMCprimo. If "**n**" is not stated, all input functions of "**g**" are inhibited. If "**t**", "**n**" and "**g**" are not stated, all input functions are inhibited. An input function can be inhibited, for example, to ensure that a program can only be started once. After the program has been called, this command is used to inhibit the input function, i.e. the command can only be repeated if the inhibit on the input function has been cancelled using the EI command. This method can also be used to inhibit input functions if they could disrupt the execution of other functions. (This command does not influence the commands "**II**" or "**WI**"!). If the state of an inhibited input changes, the respective input function will not be executed after the input is enabled with the EI command, unlike the masked input.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

Example:

PMCprimo user	Comment
0.1:	
0.1: ES PROG15	Entry of program PROG15
002E PR'PROG15 started'	Text output at program start
003E CH0.1;PC	Activate axis 1 position control
004E CH0.1;MR500	Execute relative positioning of 500 increments
005E NS	End of program entry
0.1: DI2:5+,PROG15	Determine input definition: Rising edge starts PROG15
0.1: RI2:5	Read input states
0.1:	
# Inputs 0.2	
8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0	All inputs FALSE
E E E E E E E E	Enable all input definitions
0.1: BI2:5	Disable input 2:5
0.1:	
	In the mean time, input 2:5 has changed its state, PROG15 was not executed, as input masked.
0.1:	
0.1: RI2	Read input states
0.1:	
# Inputs 0.2	
8 7 6 5 4 3 2 1	
0 0 0 1 0 0 0 0	Input 2:5 TRUE
E E E B E E E E	Input 2:5 disabled
0.1: EI2:5	Enable input 2:5
0.1:	PROG15 is not started, although input state at EI does not equal that at MI
0.1:	

3.11 Digital input and output commands

EI[t.][g:n] Enable function input

(**E**nable function **i**nput)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user enables a defined input function or input group, that of the input defined with “n” of input byte address “g”, in PMCprimo. Input functions can be masked with the MI command (see Page 3-144) or inhibited with the BI command (see Page 3-146). If “n” is not stated, all input functions of “g” are enabled. If “g” is not stated, PMCprimo automatically selects Byte 1. If “n” and “g” are not stated, all input functions are enabled.

EN[t.]g:n,bb Enable function input with priority

(**E**nable function i**n**put with priority)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8
Permitted values for bb:	0000 0000 to 1111 1111

The bit mask “**bb**” indicates which inputs are enabled. If several programs are pending execution due to a previous masking (“**MI**” command, Page 3-144), the program belonging to the lowest input number will be started. The rest will be rejected (see also “**EI**” command, Page 3-148).

UIt.g:n Undefine input definition

(**U**ndefine **i**nput definition)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8

With this command, the user deletes a defined input function, that of the input defined with “**n**” of input byte “**g**”, in PMCprimo. This command deletes an existing input function, irrespective of which input function was previously defined. “**t**”, “**n**” and “**g**” must always be stated for the UI command.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

UOt.g:n Undefine output definition

(**U**ndefine **o**utput definition)

Permitted values for t: 0 to 150 (depending on the addresses of the network subscribers) or
no details
Permitted values for g: 1 to 8 (PMCprimo C)
Permitted values for n: 1 to 8

With this command, the user deletes a defined output function, that of the input defined with "**n**" of output byte "**g**", in PMCprimo. This command deletes an existing output function, irrespective of which output function was previously defined. "**t**", "**n**" and "**g**" must always be stated for the UO command.

TCg:n[±,X1,X2] Define timer/counter output

(Define **t**imer/**c**ounter output)

Permitted values for g: 1 to 8 Permitted values for n: 1 to 8
Permitted counter values X1: 0 to 65535 for modes 0 to 7 and 0 to 31 for mode 8
Permitted modes X2: 0 to 8

This command can be used to define a digital output of Byte "**g**" for a timer or counter function. Once the output address and signal level sign ± have been stated, two further parameters must be stated. The first parameter is the timer/counter value and the second value is the required mode.

The following modes can be selected:

Mode	Function
0	One-shot up counter
1	Cyclical up counter
2	One-shot down counter
3	Cyclical down counter
4	One-shot timer
5	Cyclical timer
6	One-shot timer with restart
7	Pulse edge extension
8	Shift register

3.11 Digital input and output commands

The output is set to true (selected with ±) when the output is triggered for the first time and the output is set to false when the specified timer or counter value is reached.

In all counter modes, the counter is incremented or decremented each time that the output g:n is set with the "**SO**" command (set output) or an output function sets the output "**g**":"**n**" to true. If the up counter mode is selected, the counting process begins with 0 and ends with the stated counter value. In down counter mode, the counting process begins with the stated value and ends at 0. At the first counting edge, the output is set to true and the counter is set to its start value. If the counter reaches its end value, the output and the counter are reset. The counter can be stopped and reset at any time using the CO command.

In the counter modes, the timer is started in exactly the same way as the counters; however, each millisecond of the timer value is incremented until the specified end value is reached.

The output g:n is set to true at the first trigger signal and is reset to false once the time has elapsed.

In one-shot modes, the timers and counters operate as described above. If the timer/counter reaches its end value in cyclical modes, the output g:n changes state; the timer/counter is set to its start value and starts again.

Mode 6: One-shot timer with restart.
If the defined output is set with SOn:m while the timer is already running, the timer will start to count again. The output remains constantly set.

In shift register mode 8, the counter operates as a shift register up to a maximum value of 31 Bits. The state of the TG gate input belonging to this shift register is copied into the shift register when the mapped signal change occurs at the TK test pulse input or with an SO command at the "TC output". At the same time, the "TC output" is set to the value of the last bit of the shift register. The TZ command or the CO command for the "TC output" clear the shift register.

In mode 7, the stated edge is extended by X1 x 1ms (see Example 5). This function can be used to extend pulses of output functions.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

Example 1:

PMCprimo user	Comment
`0.2>  TC1:3+,20,1<CR>`	This command line is used to define a cyclical up counter at output 3 of Byte 1. The output switches its signal level after 20 counter pulses. The SO1:3 command increments the counter and the CO1:3 command stops the counter and resets it.

Example 2:

PMCprimo user	Comment
`0.2>  RA1:4+`	Pulsed output defined at valid reference input signal
`0.2>  TC1:4+,5,4<CR>`	This command line is used to add a one-shot timer to this output. The timer is started each time a valid reference input signal is detected and this pulse is extended to 20 ms. This extends the normal pulse time of 1 ms, making it easier for a PLC to detect the signal.

Example 3:

PMCprimo user	Comment
`2>  LO1<CR>`	Display the output definitions made in Examples 1 and 2

```
# Outputs 0.1
0.1:1
0.1:2
0.1:3 + TC Count=00020 Mode=1
0.1:4 + RA1 -> TC Count=00005 Mode=4
0.1:5
0.1:6
0.1:7
0.1:8
```

Example 4:

PMCprimo user		Comment
`0.2>`	`TC1:5+,10,8<CR>`	This command line is used to define a 10 bit shift register at output 1:5.
`0.2>`	`TG2:1+,1,5<CR>`	This command line is used to define input 2:1 as a gate input of the shift register from output 1:5.
`0.2>`	`TC2:2+,1,5<CR>`	This command line is used to define input 2:2 as a test pulse input of the shift register from output 1:5.

Example 5:

PMCprimo user		Comment
`0.2>`	`TC1:1+,10,7<CR>`	This command line is used to extend the rising edge at output 1:1 by 10ms.
`0.2>`	`so1:1;wt5;co1:1`	Output 1:1 is set and is to be reset after 5ms. However, as the TC7 command is active, the output is only reset after 15ms.

LC[g:n] List counter value

(**L**ist **c**ounter value)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

This command can be used to list the counter status of the timer/counter from output g:n. The value of the shift register is listed as a binary number. It is possible to read the counter status into a variable.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.11 Digital input and output commands

TKg:n[±,g2:n2] **Define timer/counter clock input**

(Define **t**imer/counter cloc**k** input)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8
Permitted values for g2: 1 to 8
Permitted values for n2: 1 to 8

This command can be used to link a digital output of Byte "**g**" with the output g2:n2 of a defined timer or counter function. Once the input address and signal level sign ± have been stated, the timer/counter output must be stated with g2:n2.

If the state of a counter input switches from false to true, the corresponding counter is incremented or decremented, as with the "**SO**" command (see also "**TC**" command, Page 3-149). If the state of a reset input (defined with the TZ command) switches from false to true, the timer/counter is stopped and reset as with the CO command. If an enable input has been defined with the "**TG**" command, a timer/counter will only be incremented if the enable input is true. If the signal level of the enable input is false, the timer/counter will not be incremented/decremented.

Example:

PMCprimo user	Comment
`0.2>  TK1:2+,1:1<CR>`	This command line is used to link input 1:2 to the timer/counter of output 1:1.
`0.2>  LI1`	Display of input function Byte 1 gives:
`1:1`	
`1:2 + TK1:1`	
`1:3`	
`1:4`	
`1:5`	
`1:6`	
`1:7`	
`1:8`	

3.11 Digital input and output commands

TGg:n[±,g2:n2] **Define timer/counter gate input**

(Define **t**imer/counter **g**ate input)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8
Permitted values for g2: 1 to 8
Permitted values for n2: 1 to 8

This command can be used to link a digital input of Byte "**g**" with the output. Once the input address and signal level sign ± have been stated, the output must be stated with g2:n2.

If the output is defined as a cam, this will only be set if the enable input is true.

If the output is defined as a timer or counter, counting will only take place if the enable input is true.

The "**TG**" command can enable several outputs simultaneously with one input. To do this, "**TG**" is mapped to the outputs and is displayed with the "**LO**" command (see Page 3-181). To undo the definition, the corresponding output must be stated (TG g2:n2).

TZg:n[±,g2:n2] **Define timer/counter zero input**

(Define **t**imer/counter **z**ero input)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8
Permitted values for g2: 1 to 8
Permitted values for n2: 1 to 8

This command can be used to link a digital input of Byte "**g**" with the output of a defined timer or counter function. Once the input address and signal level sign ± have been stated, the timer/counter output must be stated with g2:n2.

If the state of a reset input switches from false to true, the timer/counter is stopped and reset as with the CO command (see "**TK**" command, Page 3-153).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.12 Analogue input and output commands

3.12 Analogue input and output commands

Querying and linking of analogue inputs and setting / resetting of analogue outputs. PMCprimo C itself does not have any analogue inputs/outputs, but it can link into the analogue inputs/outputs from the CAN-IOs / PMCprotego or PMCtendoDD5.

The analogue inputs and outputs are automatically mapped to the axes in sequence.

FIt,n Free analogue input

(**f**ree analogue **i**nput)

Permitted values for t: 1 to 150 (1-49 CANopen, 51-150 SafetyNet)
Permitted values for n: 0 to 1

This command can be used to enable or mask all the analogue inputs on a network subscriber. If n==1, the inputs are enabled; if n ==0, they are masked. If n is not stated, the current state of the inputs on the network subscriber is displayed.

FOt,n Free analogue output

(**f**ree analogue **o**utput)

Permitted values for t: 1 to 150 (1-49 CANopen, 51-150 SafetyNet)
Permitted values for n: 0 to 1

This command can be used to enable or disable all the analogue outputs on a network subscriber. If n==1, the inputs are enabled; if n ==0, they are masked.

If n is not stated, the current state of the outputs on the network subscriber is displayed.

SFn Set monitor output function

(Set monitor output function)

Permitted values for nn: 0 to 15
Factory setting: 0

This command can be used to set the output function for the analogue auxiliary output. This makes it easier to set the control parameters for the respective axis, for example. A currently unused analogue output is used to output the analogue values with the AO command.

If the control point changes by 1, this corresponds to a voltage change of 4.88 mV.

3.12 Analogue input and output commands

nn	Function	Rel. command
0	No function	
1	Demand velocity Control point = (SetVel * KM * 15) / 1000000+ OM	
2	Actual velocity Control point = (ActVel * KM * 15) / 1000000+ OM	
3	Position error Control point = (FE * KM * 4) / 1000 + OM	FE
4	Velocity error Control point = (ActVel - SetVel* KM * 15) / 1000000+ OM	
5	Absolute demand position Control point = (DD * KM * 4) / 1000 + OM	DD
6	Absolute actual position Control point = (DP * KM * 4) / 1000 + OM	DP
7	Average actual velocity Control point = (DV * KM * 15) / 1000000+ OM	DV
8	Velocity of master axis Control point = (MasterVel * KM * 15) / 1000000+ OM	
9	Velocity differential master axis Control point = (DiffMasterVel * KM * 15) / 1000000+ OM	
10	Setpoint web tension/torque control Control point = AC+ OM	AC
11	Reference error Control point = (DF * KM * 4) / 1000 + OM	DF
12	Encoder snapshot Control point = (DS * KM * 4) / 1000 + OM	DS
13	Demand velocity including reference error correction Control point = (Vel * KM * 15) / 1000000+ OM	
14	Velocity ratio slave/master. The average velocities (“**VT**” Page 3-201 and “**BT**” Page 3-202) are calculated. The ratio is calculated as follows: Control point = DV (Slave) * KM / DV (Master) + OM	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.12 Analogue input and output commands

KM±nn Set monitor output gain

(Set **m**onitor output gain)

Permitted values for nn: ±65 535
Factory setting: 1

The analogue auxiliary output is not influenced by the gains of the control algorithm. The analogue auxiliary output is independent of the control parameter presettings (CW command, Page 3-54). Before the value at the analogue auxiliary output is issued, the value is multiplied by nn. The output signal can be inverted by changing the prefix.

OM±nn Set monitor output offset

(Set **m**onitor **o**utput offset)

Permitted values for n: ±32,767VIK
Factory setting: 0

This command can be used to provide the analogue auxiliary output with a fixed voltage level (offset).

The following applies for the analogue CAN outputs:
Offset in mV = 0.305 mV * OM

Example of common settings of OM and KM:

PMCprimo user		Comment
0.1>	OM0<CR>	No offset at the analogue auxiliary output
0.1>	KM500<CR>	Set amplification factor 500 (or 1000)
0.1>		for measurement with oscilloscope.

3.12 Analogue input and output commands

Typical signal characteristics on the analogue auxiliary output with a poorly tuned control loop:

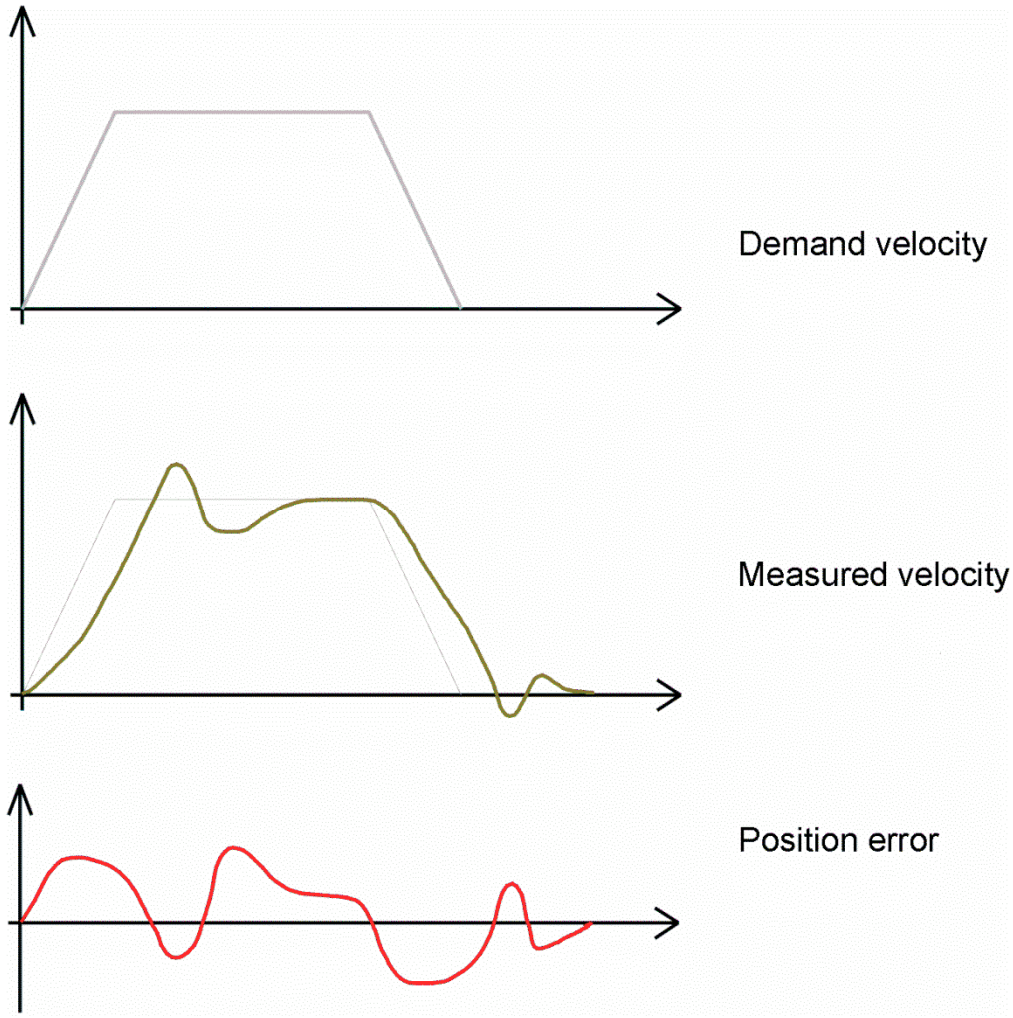


Figure 37: Monitor function on the analogue auxiliary output

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.12 Analogue input and output commands

AO[n] Set auxiliary output channel

(Set **a**uxiliary **o**utput channel)

Permitted values for n: 0 to number of available analogue outputs
Factory setting: 0

This command can be used to define an analogue output as an analogue auxiliary output using "**n≠0**". The function of the analogue auxiliary output is selected on the required axis using the SF command (Page 3-115).
The analogue auxiliary output function can only be selected on an axis on which the position control loop is open (MO command) or which is operated virtually (VM command, Page 3-32). Voltage is output at the analogue output. After a function has been selected with "**SF≠0**", the axis cannot leave virtual mode and the position control loop cannot be closed with "**PC**".

The definition of an analogue auxiliary output can be rescinded on the relevant axis by specifying "**n=0**".
A variable can now be written with the current value of the analogue auxiliary output.

With the setting FS19 to 26 (Page 3-182), the analogue output is not needed for position control. In this case it is possible to use the "**AO**" command also while position control is active ("**PC**" command).

Example:

PMCprimo user		Comment
0.1:	AO 0.2<CR>	Definition of analogue auxiliary output
0.1:	$VAR=AO<CR>	Definition is transferred to variable
0.1:	$VAR<CR>	Value query
0.1:		
0.1:	$VAR=2 (0x00000002)	

3.12 Analogue input and output commands

DA Display analogue input

(**D**isplay **a**nalogue input)

This command can be used to display the current value of the analogue input. The value range is ±2047. This corresponds to +/- 10V.

CAn Copy analogue input

(**C**opy **a**nalogue input)

Permitted values for n: 1 to the number of analogue inputs in the network

With the CA command, an analogue input can also be read on another channel. As a result, the analogue input is also available to other axes, for web tension control for example. To read the second analogue input on axis 4, for example, you will need to enter CH0.4;CA 2.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

Definition of digital input and output functions, such as "Input is software limit switch" or "Assign input definition to input". These definitions are axis-related.

The digital inputs and outputs are divided into groups of 8 (=Byte). Each address must be provided with a byte address. A colon is used to separate the byte address (g) and bit address (n). If the IOs are to be addressed by a network subscriber, the network subscriber number (t) must also be stated. If they are not stated, the IOs on the PMCprimo are addressed.

Examples:

PMCprimo user	Comment
`0.1: SO1:2`	Set output 2 of Byte 1 on network subscriber 0.
`0.1: SO1.1`	Set all unused outputs of Byte 1 on network subscriber 1.
`3.1: CO2:5`	Reset output 5 of Byte 2 on network subscriber 3.
`0.1: RI2.1:1`	Read state of input 1 of Byte 1 on network subscriber 2.
`0.1: RI2`	Read the states of all inputs of Byte 2 on the PMCprimo C.
`0.1: MI3:7`	Mask input 7 of Byte 3.
`0.1: EI4`	Enable all inputs of Byte 4 of the PMCprimo C.

3.13 Definition commands

DL g:n±[,±] Define limit switch input

(**D**efine **l**imit switch input)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

This command can be used to define a digital input as a limit switch. The definition only applies on the selected axis. The sign ± defines the status that PMCprimo evaluates as passing the end position. If the sign ± is not stated, the definition as limit switch is invalidated. If PMCprimo detects that the axis has passed a limit switch, the motor is shut down immediately.

Example:

PMCprimo user	Comment
`0.1>  CH3;DL3:7-<CR>`	This command line is used to define input 7 of Byte 3 as the limit switch of axis 3.
`0.1>`	PMCprimo evaluates the signal state 0 as the limit switch having been passed.

A direction of rotation can be added as an option. This means that the axis can again be moved away from the limit switch, without undefining it first. So monitoring only takes place in one direction of rotation.
Example:
DL 0.1:1+,+
,+ or ,- is used to limit monitoring to a positive or negative direction.
If ,+ or ,- is not stated, then the function has also been modified. Previously the axis could no longer be switched on while the limit switch was active. Now the system stores the direction of rotation at the time of switching and allows a motion task in the opposite direction.
If the axis is stationary when the limit switch is switched, then a switch-on will no longer be possible, as before.

Error message:

```
0.1> vc
0.1>
0.1V
0.1:
0.1 The software end limit has been passed in the plus direction!
(3 days 00:15:48)
0.1:
```


Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

PS g:n± **Define position snapshot input**

(Define **p**osition **s**napshot input)

Permitted values for g: 1
Permitted values for n: 1 to 6

This command can be used to store an encoder snapshot with a digital input. The stored value can be read using the DS command (see Page 3-188). The encoder snapshot is defined for the axis that is currently active (prompt). Only fast inputs can be used for the definition. The byte address "**g**" must be stated. The encoder snapshot is only executed if this function has previously been activated using the RM command. If ± is not stated, an existing definition is invalidated. The sign + or - indicates the direction of movement during which the encoder snapshot is to be executed. The stored value for the encoder snapshot is independent of a reference offset set using "**RF**".

If a reference signal has already been defined for the axis ("**DR**" or "**DZ**" command), an encoder snapshot cannot also be defined on this axis.

3.13 Definition commands

Example 1:

PMCprimo user	Comment
`0.2:  CH4;PS1:2+<CR>`	With this command line, input 2 of Byte 1 is used to execute an
`0.4:`	encoder snapshot on axis 4.
`0.4:  CH4;$VAR=DS<CR>`	The measured value from the position counter on axis 4 can be read into the VAR variable to enable the encoder snapshot to be processed further.

Example 2:

PMCprimo user	Comment
`4:  LI<CR>`	Read input definitions Byte 1.
`# Inputs 0.1`	
`0.1:1  undef`	
`0.1:2 + PS4`	Input 2 is encoder snapshot of axis 4.
`0.1:3  undef`	
`0.1:4  undef`	
`0.1:5  undef`	
`0.1:6  undef`	
`0.1:7  undef`	
`0.1:8  undef`	
`0.4:`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

DI[t.] g:n±,Name **Define function input**

(**D**efine function **i**nput)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for n:	1 to 8
Program name:	Name of a program

This command can be used to assign a program to a digital input. If the stated program is called although it is not defined, an error message is issued. The ± sign indicates the direction of the pulse edge at which PMCprimo calls the program. A separate program can be assigned to each pulse edge, provided both pulse edges are defined as input functions.

If no program name is stated, the definition of the input function ("+" or "-") is invalidated.

Example 1:

PMCprimo user	Comment
`0.2>  DI2:2-,TestPrg<CR>`	With this command line, input 2 of Byte 2 is used to call the program "**TestPrg**" at a falling edge.

A program must be run through completely before it is called again (PMCprimo error message: Cannot start the program <....> because it is already running)!

3.13 Definition commands

MG[t.] g,bb Define input mask group

(Define input **m**ask **g**roup)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for bb:	0000 0000 to 1111 1111 (default setting: 0)

With this command the user can define an input mask in PMCprimo via the inputs at the axis selected using the CH command. An input mask can be defined, for example, to ensure that conflicting commands that may not or cannot be executed simultaneously are mutually interlocked. All other inputs on this input mask are disabled with a pulse edge from an input on this input mask. The disabled inputs can only be re-enabled using the EI command. The "**II**" and "**WI**" commands are not influenced by the MG command. The right-hand bit stands for input 1. Each bit stands for the respective input. An input mask can only be defined for inputs with the same byte address. If "**bb**" is not stated, the input mask that is currently defined will be displayed.

If the state of a masked input changes, the respective input function will be executed after the input is enabled (see also MI command, Page 3-144).

Example:

PMCprimo user		Comment
0.1>	ES PROG1	
002E	CH1;VC+	Travel in + direction
003E	NS	
0.1>	ES PROG2	
002E	CH1;VC-	Travel in – direction
003E	NS	
0.1>	CH1;DI2:1+;PROG1<CR>	Call program 1
0.1>	CH1;DI2:2+;PROG2<CR>	Call program 2
0.1>	MG2,00000011<CR>	Input mask on inputs 1 and 2, Byte 2
0.1>		

The pulse edge from input 1 masks input 2. The pulse edge from input 2 masks input 1. Without an input mask, a travel command can occur in the opposite direction. Without an input mask, the axis changes its direction of travel. With an input mask, there is no reaction from PMCprimo.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

BG[t.] g:bb Define input inhibit group

(Define input inhi**b**it **g**roup)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for bb:	0000 0000 to 1111 1111 (default setting: 0)

With this command the user can define an input inhibit on the inputs in PMCprimo. An input inhibit can be defined, for example, to ensure that conflicting commands that may not or cannot be executed simultaneously are mutually interlocked. All other input functions on this input mask are inhibited with a pulse edge from an input on this input mask. The inhibited input functions can only be re-enabled using the EI command. The "**II**" and "**WI**" commands are not influenced by the BG command.

If the state of an inhibited input changes, the respective input function will not be executed after the input is enabled, in contrast to the MG command (see also MI command, Page 3-144).

Example:

PMCprimo user		Comment
0.1>	ES PROG1	
002E	CH1;VC+	Travel in + direction
003E	NS	
0.1>	ES PROG2	
002E	CH1;VC-	Travel in – direction
003E	NS	
0.1>	CH1;DI2:1+;PROG1<CR>	Call program 1
0.1>	CH1;DI2:2+;PROG2<CR>	Call program 2
0.1>	BG2:11<CR>	Input inhibit on inputs 1 and 2, Byte 2
0.1>		

The rising edge from input 1 inhibits input 2. The rising edge from input 2 inhibits input 1. Without an input inhibit, a travel command can occur in the opposite direction. Without an input inhibit, the axis changes its direction of travel. With an input inhibit, there is no reaction from PMCprimo.

3.13 Definition commands

MN[t.] g,bb **Define masked input line**

(Define **m**asked i**n**put line)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for bb:	0000 0000 to 1111 1111 (default setting: 0)

With this command the user can define an input mask for an input group in PMCprimo. In contrast to the MG command (Page 3-165), only the addressed input is masked. The masked input can only be re-enabled using the EI command. The right-hand bit stands for input 1. Each bit stands for the respective input. If "**n**" is not stated, the input mask that is currently defined will be displayed.

If the state of a masked input changes, the respective input function will be executed after the input is enabled (see also MI command, Page 3-144).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

Example:

PMCprimo user	Comment
`0.1: ES Prg1`	Program Prg1 is defined, which outputs the parameter value
`002E CH0.1;sv`	"**SV**" for test purposes.
`003E NS`	
`0.1:`	
`0.1: ES Prg2`	Program Prg2 is defined, which outputs the parameter value
`002E CH0.1;SA`	"**SA**" for test purposes.
`003E NS`	
`0.1:`	
`0.1: DI0.3:1+,Prg1`	For virtual input 3:1, Prg1 is defined as the input function for the rising edge.
`0.1: DI0.3:2+,Prg2`	For virtual input 3:2, Prg2 is defined as the input function for the rising edge.
`0.1: EI`	All inputs are enabled.
`0.1: RI0.3`	Input group 3 is displayed:
`0.1:`	
`# Inputs 0.3`	
`8 7 6 5 4 3 2 1`	
`0 0 0 0 0 0 0 0`	All inputs are reset.
`E E E E E E E E`	All inputs are enabled.
`0.1:`	
`0.1: MN0.3,11111111`	Each input should be masked automatically after it is called.
`0.1: SO3:1;WT10;CO3:1`	By setting and resetting the virtual output, a signal is generated for the virtual input.
`0.1:`	
`0.1 SV=20000`	Program Prg1 has therefore been executed.
`0.1: RI0.3`	Input group 3 is displayed:
`0.1:`	
`# Inputs 0.3`	
`8 7 6 5 4 3 2 1`	
`0 0 0 0 0 0 0 0`	
`E E E E E E E M`	Input 3:1 has been masked due to the MN command.
`0.1: SO3:1`	A further input signal has no effect.
`0.1: EI`	Only the enable triggers the program again.
`0.1:`	
`0.1 SV=20000`	Program Prg1 has been triggered via EI.

3.13 Definition commands

BN[t.] g,bb Define input inhibit line

(Define input inhi**b**it li**n**e)

Permitted values for t:	0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g:	1 to 8 (PMCprimo C)
Permitted values for bb:	0000 0000 to 1111 1111 (default setting: 0)

With this command the user can define an input inhibit for an input group in PMCprimo. In contrast to the BG command (Page 3-166), only the addressed input is inhibited. The inhibited input can only be re-enabled using the EI command. The right-hand bit stands for input 1. Each bit stands for the respective input. If "**n**" is not stated, the input inhibit that is currently defined will be displayed.

If the state of an inhibited input changes, the respective input function will not be executed after the input is enabled, in contrast to the MN command (see also MI command, Page 3-144).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

Example:

PMCprimo user	Comment
`0.1: ES Prg1`	Program Prg1 is defined, which outputs the parameter value
`002E CH0.1;SV`	"**SV**" for test purposes.
`003E NS`	
`0.1:`	
`0.1: ES Prg2`	Program Prg2 is defined, which outputs the parameter value
`002E CH0.1;SA`	"**SA**" for test purposes.
`003E NS`	
`0.1:`	
`0.1: DI0.3:1+,Prg1`	For virtual input 3:1, Prg1 is defined as the input function for the rising edge.
`0.1: DI0.3:2+,Prg2`	For virtual input 3:2, Prg2 is defined as the input function for the rising edge.
`0.1: EI`	All inputs are enabled.
`0.1: RI0.3`	Input group 3 is displayed:
`0.1:`	
`# Inputs 0.3`	
`8 7 6 5 4 3 2 1`	
`0 0 0 0 0 0 0 0`	All inputs are reset.
`E E E E E E E E`	All inputs are enabled.
`0.1:`	
`0.1: BN0.3,11111111`	Each input should be masked automatically after it is called.
`0.1: SO3:1;WT10;CO3:1`	By setting and resetting the virtual output, a signal is generated for the virtual input.
`0.1:`	
`0.1 SV=20000`	Program Prg1 has therefore been executed.
`0.1: RI0.3`	Input group 3 is displayed:
`0.1:`	
`# Inputs 0.3`	
`8 7 6 5 4 3 2 1`	
`0 0 0 0 0 0 0 0`	
`E E E E E E E B`	Input 3:1 has been masked due to the MN command.
`0.1: so3:1;wt10;co3:1`	A further input signal has no effect.
`0.1:`	

3.13 Definition commands

DE g:n± Define error output

(**D**efine **e**rror output)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

This command can be used to define a digital output as an error output. The output specified with "**n**" from Byte "**g**" carries the signal level specified with ± if PMCprimo has detected an axis error (motor off). If the position control loop is closed again (PC command), the error output is reset.

In the event of the following errors, PMCprimo switches the signal level at a defined error output:

- Position error limit exceeded (SE command)
- Monitoring time for rotary encoder signals exceeded (TO command)
- Software limit switch exceeded (DL command)
- Motor position outside the software end limits (LL or LH command)
- Errors that can be cleared with the EW command for an axis error

Example:

PMCprimo user	Comment
`0.2>  CH1;DE2:1-<CR>`	This command line is used to define output 1 of Byte 2 as an error output of axis 1 (output 2:1 carries signal level 0 in the event of an error).
`0.1:  LO2<CR>`	List output definitions Byte 2.
`# Outputs`	
`0.2:1 - DE1`	Output 1 of Byte 2 is an error output of axis 1.
`0.2:2  undef.`	
`0.2:3  undef.`	
`0.2:4  undef.`	
`0.2:5  undef.`	
`0.2:6  undef.`	
`0.2:7  undef.`	
`0.2:8  undef.`	
`0.1:  CH1;DE2:1<CR>`	This command line is used to cancel the definition of output 1 of Byte 2 as error output of axis 1.
`0.1:`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

PO g:n[±,X1,X2] Define position trigger output

(Define **p**osition trigger **o**utput)

Permitted values for g:	1 to 8
Permitted values for n:	1 to 8
Start position X1:	±2,000,000,000
End position X2:	±2,000,000,000

This command can be used to define a digital input of Byte "**g**" as an electronic cam. The output specified with "**n**" carries the signal level specified with ± within the position range specified by the prefix. If + or - is not stated, the definition is invalidated.

In the following example, PMCprimo switches the signal level between the positions 500 and 800:

Example 1:

PMCprimo user		Comment
`0.2>`	`PO2:5-,500,800<CR>`	This command line is used to define output 5 of Byte 2 as the electronic cam of axis 2
`0.2>`		with signal level 0 between the positions 500 and 800.
`0.2>`	`PO2:5<CR>`	This command line is used to cancel the above definition.
`0.2>`		

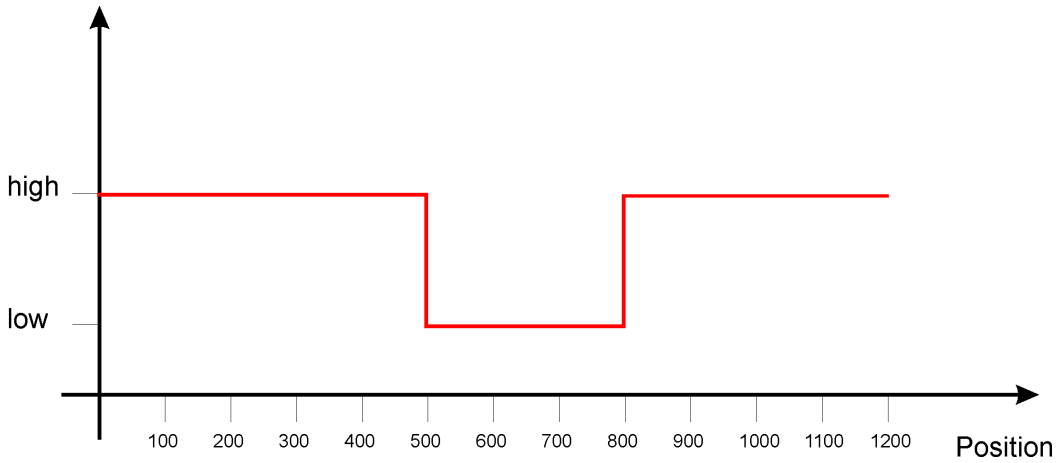


Figure 38: Example 1 for PO command

Example 2:

PMCprimo user		Comment
`0.2>`	`SB1000<CR>`	Cycle length is set to 1000.
`0.2>`	`PO2:5+,0,-200`	Set electronic cam of axis 2 with signal level 1
`0.2>`	`PC;VC-`	System travels in negative direction

In this example, switching occurs in the range between –0 and –200, whereby the cycle length is set at 1000.

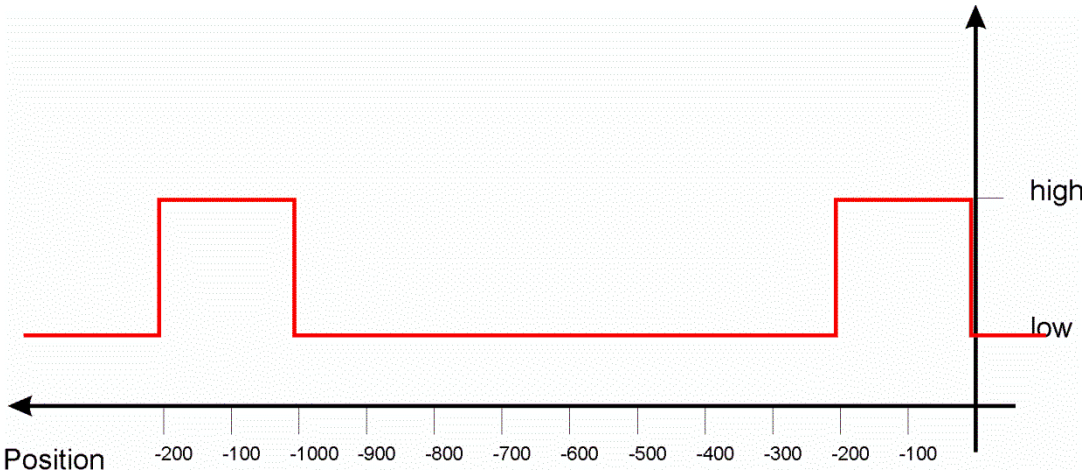


Figure 39: Example 2 for PO command

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

PHnn Define hysteresis for position trigger output

(Define **h**ysteresis for **p**osition trigger output)

Permitted values for nn: ±2,000,000,000

This command is used to extend the PO command. A hysteresis is added to the switching points, so that the output at standstill is not constantly switched on and off, if the actual position fluctuates by 1 increment, for example. The set value is either added to or subtracted from the cam position.

VO g:n[±,X1,X2] Define velocity trigger output

(Define **v**elocity trigger **o**utput)

Permitted values for g:	1 to 8
Permitted values for n:	1 to 8
Start velocity X1:	**±**40,000,000
End velocity X2:	**±**40,000,000

This command can be used to set a digital output of Byte "**g**", depending on the current velocity of an axis. The output specified with "**n**" carries the signal level specified with ± within the velocity range specified by the prefix. X1 must be <X2.

If + or - is not stated, the definition is invalidated.

In the following example, PMCprimo switches the signal level between the velocities 500 and 10000:

Example:

PMCprimo user		Comment
0.2>	VO2:5-,500,10000<CR>	This command line is used to define output "**5**" of Byte "**2**" depending on the velocity of axis 2 with signal level 0 between the velocities 500 and 10000.
0.2>		
0.2>	VO2:5<CR>	This command line is used to cancel the above definition.
0.2>		

3.13 Definition commands

VHnn Define hysteresis for velocity trigger output

(Define **h**ysterese for **v**elocity trigger output)

Permitted values for nn: ±40,000,000

This command is used to extend the VO command. A hysteresis is added to the switching points, so that the output at standstill is not constantly switched on and off, if the actual velocity fluctuates by 1 increment, for example. The set value is either added to or subtracted from the velocity switching points.

DBnn Set input debounce time

(Set input **de**b**o**unce time)

Permitted values for n: 0 to 10000
Factory setting: 0

This command can be used to set a debounce time for the real and virtual digital inputs. The values for "**n**" are stated in milliseconds. Reference inputs can be debounced with the "**RH**" command (see Page 3-134).

BO g:n± Define bound overflow output

(Define **b**ound **o**verflow output)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

This command can be used to activate a pulsed output when the cycle limit is exceeded. If the selected axis exceeds the cycle limit, PMCprimo outputs a 1 ms pulse at the output defined as "**n**" from Byte "**g**". The ± sign indicates the signal level of the output for the pulsed output. If + or - is not stated, the definition can be invalidated.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

OW g:n± Define outside window output

(Define **o**utside **w**indow output)

Permitted values for g: 1 to 8 Permitted values for n: 1 to 8

With this command, a digital output can be used to display the lag distance information larger than the target window “**SW**”. This function can be used as information: "Axis within position window" or as advance warning for reaching the position error limit “**SE**”. The character ± indicates the signal level of the output when the axis is within the position window specified with “SW”. If + or - is not stated, the definition can be invalidated.

OA g:n± Define output alignment move

(Define **o**utput **a**lignment move)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

With this command, a digital output can be used to display the information: “Alignment run active” while mapping is active (e.g. where MB, MF, SM have changed). The alignment run during the coupling process will be ignored.

The character ± indicates the signal level of the output when the axis is in an alignment run. If + or - is not stated, this output function can be invalidated.

3.13 Definition commands

RR g:n± Define reference reject output

(Define **r**eference **r**eject output)

Permitted values for g: 1 to 8 Permitted values for n: 1 to 8

This command can be used to define a digital output of Byte "**g**" as a reference error output. Each reference signal is used to measure a reference error. The measured reference error is comparerd with the value of "**SR**". If the measured reference error is within the limits specified with "**SR**", the reference error output is set to false. If the opposite is the case, i.e. the measured reference error exceeds the limits specified with "**SR**", the reference error output is set to true. The signal level of the reference error output is specified with the sign + or -. The signal level is maintained until a new comparison has been run with the next reference signal. The definition as reference error output is cancelled if the command is entered without stating + or -.

A typical application for the RR command, for example, is to control a switchpoint that distinguishes between good parts and rejects. For example, the front edge of a part on a conveyor is measured with a photo cell. The measured value can be used in conjunction with the "**SR**" value to control a switchpoint that separates good parts from rejects.

Example:

PMCprimo user	Comment
`0.1>  SR500<CR>`	Permitted limit for reference error
	`0.1> RR1:5+<CR>` Output 5, Byte 1 is the reference error output of axis 1.
`0.1>`	

If a reference signal is detected within the range SR - 500 to SR + 500, output 5 of Byte 1 will have the state 0 at the output terminal. If a reference signal outside this range is detected, output 5 will have the state 1.

If the maximum value for reference error correction (SR) has the value **0**, the reference error output is always set to **FALSE**.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

AE g:n± Define analogue limit error output

(Define analogue limit error output)

Permitted values for g: 1 to 8
Permitted values for n: 1 to 8

This command defines a digital output from PMCprimo as an error output. If the value of the analogue input exceeds one of the set limit values, the error output will change its logical state. In the event of an error, the logical state of the output is the one that has been specified with the sign ±. If the value of the analogue input returns within the permitted area, the state of the error output will also change back.

Example:

PMCprimo user	Comment
`0.2>  CH1;AE2:2-<CR>`	This command line is used to define output 2 of Byte 2 as an analogue error output of axis 1 (output 2:1 carries signal level 0 in the event of an error).
`0.1:  LO2<CR>`	List output definitions Byte 2.
`# Outputs 0.2`	
`0.2:1  undef`	
`0.2:2 - AE1`	
`0.2:3  undef`	
`0.2:4  undef`	
`0.2:5  undef`	
`0.2:6  undef`	
`0.2:7  undef`	
`0.2:8  undef`	
`0.1:  CH1;AE2:2<CR>`	This command line is used to cancel the definition of output 2 of Byte 2 as an analogue error output of axis 1.
`0.1:`	

3.13 Definition commands

LI[t.]g List input line definitions

(**L**ist **i**nput line definitions)

Permitted values for t: 0 to 150 (depending on the addresses of the network subscribers) or no details
Permitted values for g: 1 to 8 (PMCprimo C)

PMCprimo uses this command to list all the user-defined input definitions for the selected input byte.

Example:

PMCprimo user	Comment
`0.3>  LI1<CR>`	This command line is used to list all the input definitions of input byte 1.
`# Inputs 0.1`	Definitions of inputs 1 to 8
`0.1:1 - DR3`	Falling edge is reference signal from axis 3
`0.1:2 + PROG1`	Call program PROG1
`0.1:3 - TEST`	Call program TEST
`0.1:4...undef.`	Input free
`0.1:5...undef.`	Input free
`0.1:6...undef.`	Input free
`0.1:7 - DL2`	Software limit switch axis 2
`0.1:8...undef`	Input free
`0.3>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

LO[t.]g List output line definitions

(**L**ist **o**utput line definitions)

Permitted values for t: 0 to 150 (depending on the addresses of the network subscribers) or no details

Permitted values for g: 1 to 8 (PMCprimo C)

PMCprimo uses this command to list all the user-defined output definitions for the selected output byte.

Example:

PMCprimo user	Comment
`0.3>  LO1<CR>`	This command line is used to list all the output definitions for output byte 1.
`# Outputs 0.1`	Outputs 1 to 8
`0.1:1  undef`	Output free
`0.1:2  undef`	Output free
`0.1:3  undef`	Output free
`0.1:4  undef`	Output free
`0.1:5 - PO3>500<1000`	Electronic cam, axis 3: Signal level 0 from 500 to 1000 increments.
`0.1:6 + DE2`	Error output: Signal level 1 in the event of an error at axis 2.
`0.1:7 + BO4`	Pulsed output with signal level 1 at the cycle limit of axis 4.
`0.1:8 - AE1`	Analogue input limit error output, axis 1.
`0.3>`	

3.13 Definition commands

FSnn Feedback set encoder type

(**F**eedback **s**et encoder type)

Permitted values for n: 0 to 28
Factory setting: 0

This command can be used to set the required feedback system for each axis. The following feedback systems are supported:

- Incremental rotary encoder (FS 0 to 2 and 28)
- SSI rotary encoder (FS 5-8)
- CANopen rotary encoder (FS9-10)
- Transmission of set velocity values for the servo amplifier via CANopen. Position control is performed by PMCprimo (FS19-20)
- Transmission of positional set values for the servo amplifier via CANopen. Position control is performed by the servo amplifier (FS19-27)

FS	Option
0	Incremental rotary encoder, quadruple pulse evaluation
1	Incremental rotary encoder, double pulse evaluation
2	Incremental rotary encoder, single pulse evaluation
3	Reserved
4	Reserved
5	SSI rotary encoder, relative position, binary code; 300kHz
6	SSI rotary encoder, relative position, gray code; 300kHz
7	SSI rotary encoder, absolute position, binary code; 300kHz
8	SSI rotary encoder, absolute position, gray code; 300kHz
9	CANopen rotary encoder, relative position
10	CANopen rotary encoder, absolute position
11	Reserved
12	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

FS	Option
19	Servo amplifier, velocity via CANopen, relative evaluation with transmission of actual position and actual current (select corresponding servo amplifier with PD command)
20	Servo amplifier, velocity via CANopen, absolute evaluation with transmission of actual position and actual current (select corresponding servo amplifier with PD command)
21	Servo amplifier, demand position via CANopen, relative evaluation, positive reference signal (select corresponding servo amplifier with PD command)
22	Servo amplifier, demand position via CANopen, relative evaluation, negative reference signal (select corresponding servo amplifier with PD command)
23	Servo amplifier, demand position via CANopen, absolute evaluation, offset setting via RF (Page 3-125) (select corresponding servo amplifier with PD command)
24	Servo amplifier, demand position via CANopen, relative evaluation, positive reference signal with transmission of position error and actual current (select corresponding servo amplifier with PD command)
25	Servo amplifier, demand position via CANopen, relative evaluation, negative reference signal with transmission of position error and actual current (select corresponding servo amplifier with PD command)
26	Servo amplifier, demand position via CANopen, absolute evaluation, offset setting via RF (Page 3-125) with transmission of position error and actual current (select corresponding servo amplifier with PD command)
27	Servo amplifier, only demand position via CANopen; this setting is suitable for third party manufacturers that support “Interpolate Position Mode” in accordance with DS402 (select corresponding servo amplifier with PD command)
28	Incremental rotary encoder, quadruple pulse evaluation, relative position via CANopen. (select corresponding servo amplifier with PD command)

Note: If a servo amplifier is selected with PD, the following setting must be made on a selected PMCtendo DD4: “syncsrc 3” and “fpga 3”.

In the case of a PMCprotego or PMCtendoDD5, only the setting “syncsrc 3” must be made.

3.13 Definition commands

With F21 to 27, only the demand position is sent via the CANBus. Position control is performed by the servo amplifier. The position control in PMCprimo (KP, KF etc.) is not used. The latch function of the servo amplifier can be activated with FS21, 22, 24 and 25. To do this the command IN1MODE (input 1) or IN2MODE (input 2) must be set to 26 and the reference sensor must be clamped to the servo amplifier's digital input. (SAVE+COLDSTART required).

Referencing is fully supported with FS21 and 22: IN (Page 3-39), RM (Page 3-122), RW (Page 3-122) etc. (incl. FW (Page 3-128), FH (Page 3-131), FL, ZH and ZL).

The resolution can be set with the NB command (Page 3-186) (NB24 means 4096 increments per revolution). The direction of rotation is set using "**CW**" Bit 5 (Page 3-54).

Neither "**DZ**" nor "**DR**" may be set on the axis. If FS21 or FS22 is selected, no reference input from ***PMCprimo*** is assigned, but the reference recording is still activated automatically.

Note: Each time the PC command is called (Page 3-32), the current actual position is read from the servo controller and the position counter in PMCprimo is adjusted. If the axis is rotated manually, for example, the change will not be noticed until the next PC command, so no positions are lost. With relative evaluation, this is not carried out when switching on for the first time after power is applied. The current position in PMCprimo is therefore maintained. So the behaviour is identical to an incremental rotary encoder.

The selection FS 24, 25 and 26 is similar to FS21 to 23. The internal position control loop of the servo amplifier is used. The reference function with "IN1MODE 26" or "IN2MODE 26" is also possible. The lag position error and actual current are also transmitted for diagnostic purposes. To ensure that the correct value is transmitted, the drive commands "PGEARI" and "PGEARO" are automatically set to 1048576 (if "PRBASE" is 20) or 65536 (if PRBASE is 16). The maximum position error ("PRBASE") must be adapted in accordance with the required position error limit. This setting increases the bus load. So if the output of the position error and actual current is no longer needed, the setting should be switched back to FS21 to 23.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.13 Definition commands

Example 1:

Specification: PMCprimo with incremental rotary encoder (5000 pulses)

FS setting:

0:	Quadruple pulse evaluation	=>	20000 increments/revolution
1:	Double pulse evaluation	=>	10000 increments/revolution
2:	Single pulse evaluation	=>	5000 increments/revolution

Example 2:

Specification: PMCprimo with SSI rotary encoder

The resolution can be set with the NB command (Page 3-186).

FS setting:

5 or 6:	The position is evaluated relatively.
7 or 8:	The position is evaluated absolutely. The SB value (Page 3-120) must be 2^n and should not exceed the maximum encoder resolution.

Example 3:

Specification: PMCprimo networked with servo amplifier and Hiperface rotary encoder via CANopen.

The required servo amplifier must be selected using the PD command (Page 3-225).
The resolution can be set with the NB command (Page 3-186).

FS setting:

19:	The position is evaluated relatively.
20:	The position is evaluated absolutely. The SB value (Page 3-120) must be 2^n and should not exceed the maximum encoder resolution.

3.13 Definition commands

NBnn Set number of bits for SSI encoder

(Set **n**um**b**er of bits for SSI encoder)

Permitted values for nn: 12 to 32
Factory setting: 24

With this command, the number of data bits can be set for each axis when using an SSI or CANopen rotary encoder. As a result, SSI and CANopen rotary encoder types can be used, from a simple 12bit single-turn to the 24bit multi-turn model.

The following applies for transferring the position values from and to the servo amplifier (FS19 – 27):

Permitted values for nn: 12 to 32

When transferring the position values from and to the servo amplifier, the last 12 Bits of the NB value are always reserved for the number of revolutions. The remaining bits indicate the resolution per revolution.
With NB28 resolution, for example, the motor runs 65,536 increments per revolution.

FCnn Feedback change encoder

(**F**eedback **c**hange encoder)

Permitted values for nn: 0 to 60
Factory setting: 0

MSnn Master scale factor

(**m**aster **s**cale factor)

Permitted values for nn: 0 to 8
Factory setting: 0

This command can be used to scale a master value. The master position is multiplied by the factor 2^n. This only makes sense if the master position is averaged using the PT command (Page 3-53), otherwise the position changes are too great.

In conjunction with the PT command, this option increases the resolution of the measuring system. This is necessary if a slave axis with a high transmission ratio is to follow this master axis. With a ratio of 10:1, for each increment that the master moves, the slave must travel 10 increments. This will impact on the motor in terms of noise and increased temperature development. If the position is scaled and averaged, the transmission ratio is reduced, lessening the effect.

It is possible to assign a value to a variable.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

Display of data such as position, velocity, lag distance.

DP Display actual position

(**D**isplay actual **p**osition)

This command is used to display on screen the current actual position of the selected axis. The actual position is displayed in increments.

With the settings FS 21 to 26 (Page 3-182), the demand position is transmitted via the CAN bus. For example, if the axis is rotated manually (motor switched off), previously the current position could not be displayed because the actual position is not transmitted with these FS settings. Now the position is read using the "**DP**" command, so the actual position is displayed.
Note: This does not take place with the "**DM**" command (scope function). The extension only concerns the CP command.

DD Display demand position

(**D**isplay **d**emand position)

This command is used to display on screen the current demand position of the selected axis. The demand position is displayed in increments.

DV Display velocity

(**D**isplay **v**elocity)

This command is used to display on screen the current velocity of the selected axis. The actual velocity is displayed in increments/second. If the VT command is used, an average velocity over the time interval defined with VT is displayed.

FE Display following error

(Display **f**ollowing **e**rror)

This command is used to display on screen the current position error of the selected axis. This command provides the opportunity to read the current position error into a variable.

3.14 Display commands

DF Display calculated reference error

(**D**isplay calculated re**f**erence error)

With this command, the last reference error to be stored for the selected axis is displayed in increments (for reference errors see chapter entitled Reference commands, from Page 3-116).

If Bit 2 of the "**FW**" command is set, the DF value is only updated if it is within the filter limits (including "**SR**" value).

DG Display actual reference error

(**D**isplay actual re**f**erence error)

With this command, the reference error which is actually being corrected is displayed in increments. The "**DF**" command (Page 3-188), on the other hand, displays the calculated error (for reference errors see chapter entitled Reference commands, from Page 3-116).

The display of the reference error to be corrected will be updated even if Bit 0 is set to 0, in other words, the error is not corrected.

DS Display snapshot position data

(**D**isplay **s**napshot position data)

With this command, the last recorded encoder snapshot for the selected axis is displayed in increments (see PS command, Page 3-163).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

DT[n] Display time

(**D**isplay **t**ime)

This command is used to display the internal PMCprimo system time. The display format is as follows: Days:Hours:Minutes:Seconds

The DTn command has been expanded as follows:

0: Display the internal system time (see above)
1: Display the time needed by the interrupt routine, in µs
2: Display the maximum time needed by the interrupt routine since the last reset (DT3)
3: Reset the maximum time (DT2)
4: Display the current CAN bus load as %
5: Display the maximum CAN bus load since the last reset (DT6) as %
6: Reset the maximum load (DT5)
7: Display the current PLC cycle time in µs
8: Display the maximum PLC cycle time since the last reset (DT9)
9: Reset the maximum PLC cycle time (DT8)
10: Display the map generator's computing time
11: Display the processor temperature
12: Display the maximum processor temperature since cold start
13: Display the maximum processor temperature
14: Display the time since the system start in ms

DO Display mode off

(**D**isplay mode **o**ff)

This command is used to end display mode "**DM**" and trace mode "**TR**".

GM Display mapped axis bound position

(**D**isplay **m**apped axis bound position)

This command can be used to display the current cycle length. If Bit 7 of the "**MW**" command is set (Page 3-90), the slave's cycle length is automatically adapted to the position table and the current transmission ratio during active position mapping. The calculated value can be displayed using the GM command. If position mapping is not active, the SB value (Page 3-120) is displayed.

The GM value is calculated internally by the processor. The displayed value is only valid once position mapping has been activated. The GM value can only be displayed on the slave axis.

3.14 Display commands

GW Get wraparound offset value

(**G**et **w**raparound offset value)

This command can be used to display the slave offset value during master zero crossing. A misalignment due to the "**MB**" and "**MF**" commands (Page 3-88) is not considered in this case. If any drift should occur, the GW value must not be allowed to change. The automatic cycle length correction ("**MW**" command, Bit 7 Page 3-90) may be of help in this case.

The value is calculated internally by the processor. The displayed value is only valid once position mapping has been activated. The GM value can only be displayed on the slave axis.

In a simple case, when the cycle length on the master and slave axis are identical, the value must amount to the value 0 or +- cycle length. If the value changes upwards or downwards over the course of time, the axis will drift away. It is essential therefore to check the position mapping settings.

DM[nn] Continuous display mode

(Continous **d**isplay **m**ode)

Permitted values for nn: 1 to 2,000,000,000, or no details

This command is used to activate PMCprimo display mode. If "**nn**" is not stated, the display is continuous. The interval between the individual values is 100ms. The parameter 'T' in the listing displays a time counter. If you take the difference between two consecutive values, the result will be the current time base.

If n≠0 is stated, the display appears n-times on the display unit. In ths case the values are recorded in 1 millisecond intervals.

Example 1:

PMCprimo user		Comment
0.2>	DM2<CR>	This command line is used to activate display mode for 2 lines
0.2	D=0000918 P=0000918 E=+0000000 T=0000001	
0.2	D=0000920 P=0000920 E=+0000000 T=0000101	
0.2>		

The following values are listed:

D: Demand position
P: Actual position
E: Lag distance
T: Recording time sequence

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

Example 2:

PMCprimo user	Comment
`0.2>  DM<CR>`	With this command line, display mode is activated on axis 2 until "**DO**" is entered.
`0.2 D=0000918 P=0000918 E=+0000000 T=00082231`	
`0.2 D=0000918 P=0000918 E=+0000000 T=00082303`	
`0.2 D=0000918 P=0000918 E=+0000000 T=00082376`	
`0.2 D=0000918 P=0000918 E=+0000000 T=00082449DO<CR>`	
`0.2>`	

TR[nn] Enable trace mode

(**E**nable t**r**ace mode)

Permitted values nn: 0 to 10,000

This command is used to start trace mode. If nn is stated, nn values will be recorded in intervals of TT (see Page 3-192). If "**nn**" is not stated, the display will be continuous, based on the set transmission rate.

Up to 4 different pieces of information can be displayed. This function uses the oscilloscope program **PScope**. The "**TW**" command (see Page 3-192) is used to select the channel (1 to 4) and the required information. It is possible to define a trigger via the "**TH**" command (see Page 3-195) and to specify a pretrigger via the "**TP**" command (see Page 3-196).

"**TR0**" can be used to display again the last data to be recorded. However, the "**TT**" and "**TW**" commands must not be modified following the recording.

Example 1:

PMCprimo user	Comment
`0.1V  TH0`	Trigger is switched off.
`0.1V  TT4`	Time step is set to 4 milliseconds.
`0.1V  TW1,1`	Channel 1 is to record the demand position.
`0.1V  TR2`	Start the recording of 2 values.
`0.1V`	
`D0.1=00000123 T=00000004`	Demand position is 123, T indicates the time elapsed.
`D0.1=00000125 T=00000008`	
`0.1V`	

3.14 Display commands

TT[nn] **Time step for trace mode**

(**t**imestep for **t**race mode)

Permitted values nn: 0 to 10,000

This command is used to define the time step for recording with “**TR**”.

It is possible to assign a value to a variable.

TW[n,m, VarName] **Configuration word for trace mode**

(configuration **w**ord for **t**race mode)

Permitted values n:	1 to 4
Permitted values for m:	0 to 202
Permitted values for VarName:	Variable name for Type 43 and 44

It’s possible to choose between four recording channels. 'm' is used to select the required recording type. Recording of the relevant channel is deactivated with “**m=0**”:

For command type 43 and 44, the corresponding variable name must also be stated. The selected variable automatically becomes global, as it is recorded by the axis task.
The settings “**TW n, 43, $Bn**” and “**TW n, 44, $Name**” are not stored, but are displayed with “**LA**”.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

No. Type	Description	Display
m=0	No value selected	
m=1	Demand position	D
m=2	Actual position	P
m=3	Lag distance	E
m=4	Demand velocity	V
m=5	Actual velocity	W
m=6	Average actual velocity	X
m=7	Reference error	R
m=8	Encoder snapshot	S
m=9	Change of set value	U
m=10	Change of actual value	C
m=11	Value of analogue input	A
m=12	Lag distance additional control	L
m=13	Current transmission ratio	U
m=14	Position of master axis	MP
m=15	Velocity of master axis	MV
m=16	Position of differential axis	DP
m=17	Velocity of differential axis	DV
m=18	Actual current of servo controller	IS
m=20	Change of set value, absolute amount	UB
m=21	Input byte 1	I1
m=22	Input byte 2	I2
m=23	Input byte 3	I3
m=24	Input byte 4	I4

3.14 Display commands

No. Type	Description	Display
m=25	Input byte 5	I5
m=26	Input byte 6	I6
m=27	Input byte 7	I7
m=28	Input byte 8	I8
m=31	Output byte 1	O1
m=32	Output byte 2	O2
m=33	Output byte 3	O3
m=34	Output byte 4	O4
m=35	Output byte 5	O5
m=36	Output byte 6	O6
m=37	Output byte 7	O7
m=38	Output byte 8	O8
m=40	Motor fault	ME
m=41	Error variable	$F
m=42	Status variable	$S
m=43	Bus variable	$Bn
m=44	Variable	$Name
m=61	CAN input byte 1	I1
:	:	:
m=68	CAN input byte 8	I8
m=71	CAN output byte 1	O1
:	:	:
m=78	CAN output byte 8	O8
m=101	CAN input byte 1	I1
:	:	:
m=132	CAN input byte 32	I32
m=171	CAN output byte 1	I1
:	:	:
m=202	CAN output byte 32	I32

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

THnn Trigger channel for trace mode

(**t**riggerc**h**annel for trace mode)

Permitted values nn: 1 to 4

Triggering can take place on a recording channel defined with the "**TH**" command. The "**nn**" parameter indicates the required recording channel on which triggering is to take place. The trigger level is defined with the "**TL**" command; the trigger transition is defined with "**TF**".

TLnn[,m] Trigger level for trace mode

(**t**rigger**l**evel for trace mode)

Permitted values nn: ±6,000.000
Permitted values for m: 0: Trigger level signed
1: Amount for trigger level

This command defines the trigger level for recording with "**TR**". If the trigger channel defined with "**TH**" has reached the value "**TL**" with the pulse edge defined in "**TF**", recording "**TR**" is started.

It is possible to assign a value to a variable.

TFnn Trigger transition for trace mode

(**t**riggertransition for trace mode)

Permitted values nn: 0, 1 or 2

This command defines the trigger transition for recording with "**TR**". If the trigger channel defined with "**TH**" has reached the value "**TL**" with the pulse edge defined in "**TF**", recording "**TR**" is started.

The following applies when triggering on variables:
0: When the variable value changes and the variable value is less than "**TL**", recording is started.
1: When the variable value changes and the variable value is greater than "**TL**", recording is started.
2: If the variable value is changed to "**TL**", recording is started.

With all other recordings:
0: Trigger on falling edge
1: Trigger on rising edge
2: Trigger on both edges

It is possible to assign a value to a variable.

3.14 Display commands

TPnn Pretrigger for trace mode

(**p**retrigger for **t**race mode)

Permitted values nn: 0 to 10,000

If trace mode is used with the trigger “**TH**”, a pretrigger can be defined with “**TP**”. In this case, the data is stored permanently in a ring memory, so that a history of 'nn' milliseconds can also be recorded when triggering.

It is possible to assign a value to a variable.

TD[mm][,nn] Display last value of trace mode

(display last value of trace mode)

Permitted values nn:	1 to 4	Channel number
	1 to 10,000	Output line

The “**TD**” command displays the last value to be recorded on the nn channel (via the “**TR**” command). It is possible to assign a value to a variable.

The data recorded with the “**TR**” command can be displayed again without parameter data using the “**TD**” command. If “**mm**” and “**nn**” are stated, the value of channel nn is displayed from the output line “**mm**”. “**nn**” and “**mm**” may be expressions. The “**TT**” and “**TW**” commands must not be modified following the recording, so that the “**TD**” command can display the data again.

SYb Set system prompts

(Set **sy**stem prompts)

Permitted values for n: 0 to 3
Factory setting: 0

If “**SY**” is set to 1, system prompts are disabled. Also, error messages are only displayed as numbers.

The command has been expanded as follows:

SY0:	(Factory setting) All prompts displayed; variables are displayed in decimal and hexadecimal
SY1:	No prompts displayed; variables are displayed in decimal and hexadecimal
SY2:	As SY0, variables only in decimal
SY3:	As SY1, variables only in decimal
SY4:	Map generator not displayed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.14 Display commands

?? Print help display

(Print help display)

This command is used to display the main help menu:

```
Help - Main menu
================
 ??n -->  Error text for error number n
 ?1  -->  Command overview
 ?2  -->  Error and status messages on the display
 ?3  -->  Abbreviations used
 ?4  -->  Mathematical operators
 ?5  -->  Variables
 ?6  -->  Programs and tables (maps)
 ?7  -->  Map generator
 ?8  -->  Values of status variables
 ?9  -->  Number formats
```

The required help item is displayed by entering “?n”.

Help can be requested for each command by adding a “?” sign.

Example:

PMCprimo user	Comment

`0.2>  DP?<CR>`
`DP: Display actual position`
`Syntax: DP`
`0.2>`
	Help is requested for the “**DP**” command.

LE[n] Display last error

(Display **l**ast **e**rror)

Permitted values for n: 0 or 1

The following applies for n=0:
This command repeats the last 10 error messages, because under some circumstances these errors could not be evaluated at the time they occurred.

The following applies for n=1:
The last 100 error messages are displayed. As they are stored in non-volatile memory, they will not be lost when the system is switched off. Notice will also be given when the control system has re-booted. A cold start will also be entered as such, due to the watchdog.

3.14 Display commands

LGnn Set language

(Set **l**an**g**uage)

Permitted values for n: 0 to 3
Factory setting: 0

With this command the user can select the language in which PMCprimo messages are displayed in plain text.

LG	Language
0	German
1	English
2	Italian
3	French

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.15 Phase commands

Velocity-dependent phase displacement on electronic cams and with offset master position.

PA g:n,X **Set phase advance scale factor**

(Set **p**hase **a**dvance scale factor)

Permitted values for g:	1 to 8
Permitted values for n:	1 to 8
Permitted values for X:	0 to 2,000,000,000
Factory setting:	0

This command can be used to provide the electronic cam of an axis with a velocity-dependent phase displacement. The definition refers to the output g:n. To query the value, the output must also be stated.

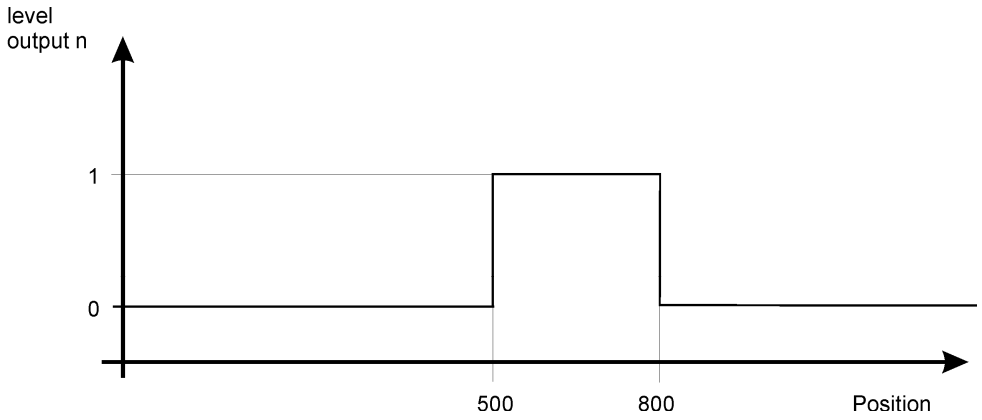
$$\text{Phase displacement output} = \left(\text{Demand velocity}/_{256}\right) \times \left(\text{PA}/_{256}\right)$$

The average demand velocity is used for the calculation. The interval of "**VT**" (Page 3-201) is used in this case, as set with the command.

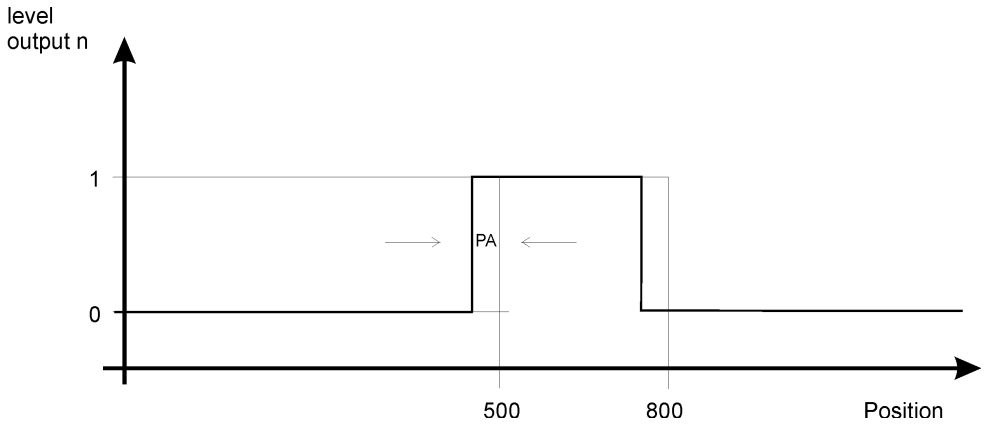
The velocity of the axis is stated in increments/second; for example, a velocity of 20,000 increments/second and a PA value of 500 results in a phase displacement of 153 increments.

3.15 Phase commands

Electric cam without velocity-dependent phase displacement:



Electric cam with velocity-dependent phase displacement at a velocity of v=1000 incr./sec. and a phase displacement of PA = 2560:



Electric cam with velocity-dependent phase displacement at a velocity of v=5000 incr./sec. and a phase displacement of PA = 2560:

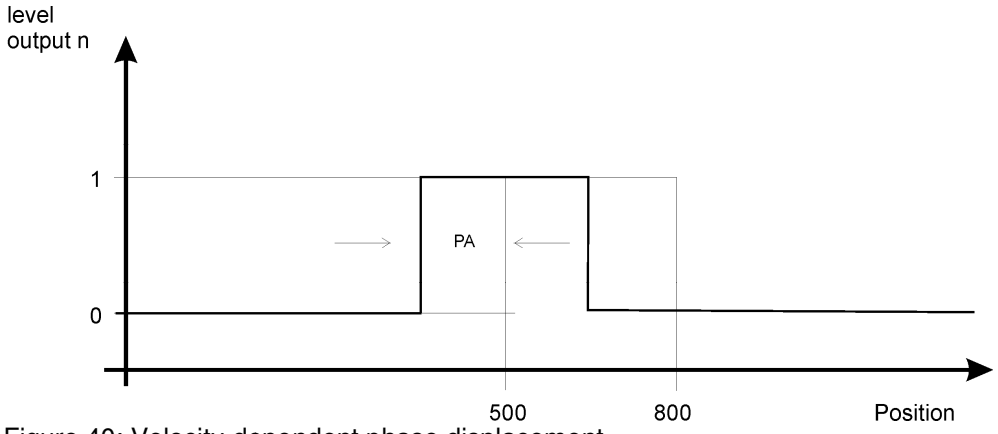


Figure 40: Velocity-dependent phase displacement

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.15 Phase commands

VTn Set velocity averaging time constant

(Set **v**elocity averaging **t**ime constant)

Permitted values for n: 1 to 10,000 milliseconds
Factory setting: 1

This command is effective when using the "**DV**" and "**PA**" command. The value "**n**" specifies a time period over which PMCprimo determines the average velocity of the selected axis. PMCprimo uses this average velocity rather than the current velocity to calculate the phase displacement on electronic cams. When n=1, PMCprimo does not determine an average velocity.

The calculated average velocity can be displayed using the DV command or can be displayed at the analogue auxiliary output. It is the actual velocity that is averaged.

BAn Set map base advance

(Set map base advance)

Permitted values for n: 0 to 2,000,000,000
Factory setting: 0

This command can be used to provide a master position offset, set using the MB command (see Page 3-87), with a velocity-dependent phase displacement.

$$\text{Phase displacement offset} = \frac{\text{Velocity of master axis}}{256} \times \frac{BA}{256}$$

The velocity of the master axis is stated in increments/second; for example, a velocity of 10,000 increments/second and a BA value of 200 results in a phase displacement of 30 increments.

The velocity of the master axis can be averaged over time. The "**BT**" command is available for this; it can be used to set the time over which the average velocity will be determined.

This factor can also be considered as a set time. A value of 1 corresponds to 15,625 microseconds. So a phase displacement of 64 corresponds to 1 millisecond and therefore the distance that the master travels in this time.

3.15 Phase commands

BTn Set base advance time constant

(Set base advance time constant)

Permitted values for n: 1 to 10,000 milliseconds
Factory setting: 1

This command is effective when using the BA command. The value "**n**" specifies a time period over which PMCprimo determines the average velocity of the axis. PMCprimo uses this average velocity rather than the current velocity to calculate the phase displacement of a master position offset on the slave axis. When n=1, PMCprimo does not determine an average velocity.

The averaged velocity is also used to determine whether the master is moving in the event of an "**XM**" command (see Page 3-82).

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.16 Web tension control with analogue input

The value of the analogue input (from the servo amplifier / CAN-IOs) is a key factor for the web tension control loop. As a result, the value of the analogue input is downloaded at regular intervals (CAN cycle time) and this is used to calculate a transmission ratio between the master and the slave axis. The algorithm for this calculation is:

$$i = SM + SM \times [AP\ e_j + AI\ \Sigma e_j + AD\ (e_j - e_{j-1})]$$

i	=	Master/slave transmission ratio
SM	=	Set transmission ratio (SM command)
AP	=	Proportional gain
AI	=	Integral gain
AD	=	Differential gain
e_i	=	Web tension distance (= set web tension – actual web tension)

The dynamic behaviour of the web tension depends on these constant gains and on the mechanical behaviour of the driven machine. These gains must be set in order to achieve optimum control behaviour.

3.16 Web tension control with analogue input

AMn Set analogue control mode

(Set **a**nalogue control **m**ode)

Permitted values for n: 0 and 1
Factory setting: 0

This command enables analogue control. "**AM1**" means enable web tension control; "**AM0**" disables analogue control. Analogue control can be enabled or disabled at any time. The value of "**AM1**" cannot be saved with the SP command. Remedy: Write the line "**AM0**" or "**AM1**" into a start-up program (see AS command, Page 3-68).

Activating position mapping with the XM command triggers various actions on the slave axis. When web tension control is enabled, these actions are defined via the web tension presetting, using the AW command. When web tension control is enabled, Bits 0, 1 and 4 of the MW command are compulsorily set to 1. This is essential for the function of the software coupling and the velocity ratio in the web tension control loop.

If Bit 4 or Bit 5 of the AW command is set to 1, PMCprimo initialises the transmission ratio automatically, before position mapping for web tension control is activated. PMCprimo measures the distance between the two positions, where the limit values for the analogue input are above or below the limit values, by moving the master/slave axis. Transmission with these measured values provides a good estimate for the transmission ratio when web tension control is activated. This value enables PMCprimo to achieve the transmission ratio much quicker than if it was using the preset SM value, especially when the machine is not starting out from its start position.

An example for this application is winding or unwinding. Normally the machine starts with a full or empty coil and the initial transmission ratio is specified by the SM value. Any restart following an interruption is a soft start, as the start uses the last transmission ratio to be determined. However, if the machine is started with partly wound or unwound coils with unknown diameters, the SM value that is normally specified will be unsuitable for the new coil diameter. Through this automatic initialisation of the transmission ratio at the start of web tension control, the machine can be started without the web tension control loop swinging greatly from the preset SM value for the currently required transmission ratio.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.16 Web tension control with analogue input

APnn Set analogue control proportional gain

(Set **a**nalogue control **p**roportional gain)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 10,000

This command is used to set the proportional gain of the control algorithm for analogue control. Setting a high value for the AP gain enables a short reaction time and precise analogue control. The AP gain should therefore be set as high as possible, without causing an overshoot.

AInn Set analogue integral control gain

(Set **a**nalogue **i**ntegral control gain)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

This command is used to set the integral gain of the control algorithm for analogue control. When using an integral gain, PMCprimo integrates the control deviation into a continuous overall error by adding the current error. The integral gain is mainly used in winding and unwinding applications with web tension control.

ADnn Set analogue differential control gain

(Set **a**nalogue **d**ifferential control gain)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 0

This command is used to set the differential gain of the control algorithm for analogue control. This gain uses the differential of the actual value distance that represents the system's control deviation. This gain is useful where there are very rapid changes in the control deviation. It has a dampening effect on the analogue control.

ACnn Set analogue control setpoint

(Set **a**nalogue **c**ontrol setpoint)

Permitted values for n: ±2,000,000,000
Factory setting: 0

This command is used to specify the setpoint value of the analogue control. The control deviation is calculated from the difference between this setpoint value and the actual value measured with the analogue input. The setpoint value of the analogue control can be incremented/decremented with the IP command, if the IS command has been set to 2.

3.16 Web tension control with analogue input

AWbb Set analogue control options word

(Set **a**nalogue control options **w**ord)

Permitted values for bb: 0 to 1111.1111
Factory setting: 0

The wildcard 'X' can be stated for each bit. This means that this bit will not be changed.

This command is used to define the behaviour of the analogue control. (Bit 0 is on the right with input/output and on the left with Bit 7).

Bit 0: This bit determines the behaviour of the slave axis when a position mapping is activated using the XM command.

0: The slave axis immediately switches to the condition of the active position mapping. If "**CT**" software coupling is used, it switches to the transmission ratio specified through the web tension.

1: The slave axis travels at jog speed (set with "**SS**") until the web tension has reached the web tension setpoint value, then it switches to the condition of active position mapping.

Bit 1: Reserved.

Bit 2: This bit determines the mode of operation of the integral gain. The web tension depends on the movement of the master and slave axis, as the output of the control loop is the velocity ratio between the two axes. If the master axis is stationary, the slave axis is also stationary and it is possible to control the velocity ratio. In this case it is necessary to block the I-section to prevent a large web tension distance from accumulating, causing a step response when starting the motor.

0: The integral gain is disabled.

1: The integral gain is enabled.

Bit 3: This bit determines the transmission ratio with which web tension control will be activated.

0: The initial transmission ratio is the SM value.

1: The initial transmission ratio is the current transmission ratio. Normally this is the last current transmission ratio determined before the slave was stopped.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Bit 4: This bit enables automatic initialisation between the upper and lower limit value on the slave axis when a position mapping is activated with web tension control (for detailed information see also “**AR**” and “**XR**”).

0: Automatic initialisation of the transmission ratio with the upper and lower limit value is not enabled on the slave axis.

1: Automatic initialisation of the transmission ratio with the upper and lower limit value is enabled on the slave axis.

Bit 5: This bit enables automatic initialisation between the upper and lower limit value on the master axis when a position mapping is activated with web tension control (for detailed information see also “**AR**” and “**XR**”).

0: Automatic initialisation of the transmission ratio with the upper and lower limit value is not enabled on the master axis.

1: Automatic initialisation of the transmission ratio with the upper and lower limit value is enabled on the master axis.

Bit 6: This bit determines the approach direction of the web tension setpoint value, if Bit 0 is set to 1 and web tension control is enabled.

0: The approach direction is the same as the sign for the web tension distance.

1: The approach direction is contrary to the sign for the web tension distance.

Bit 7: This bit determines the sign of the analogue control.

0: An increase in the control deviation (web tension distance) produces a voltage rise at the setpoint output.

1: An increase in the control deviation (web tension distance) produces a voltage reduction at the setpoint output.

ATnn Define analogue input type

(Define **a**nalogue input **t**ype)

Permitted values for nn: 0 to 2
Factory setting: 0

It is possible to select between various input values for the web tension control:

AT0: Analogue input value is DA value
AT1: Reference error is DA value
AT2: Motor current is DA value
The current value is only available with FS19 or FS20.
AT3: Analogue input value is DG value
AT101..120: The analogue input value is the actual position (DP value) of the axis nn-100.

3.16 Web tension control with analogue input

ARnn Define analogue range distance

(Define **a**nalogue **r**ange distance)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 256

This command defines or displays the distance between the upper and lower limit of the analogue input of the currently selected axis, called the analogue distance. It enables the value for the analogue distance to be specified for a master axis not controlled by PMCprimo or enables the value determined using the XR command to be displayed.

MMnn Define master axis analogue range distance

(Define **m**aster axis analogue range distance)

Permitted values for nn: 0 to 2,000,000,000
Factory setting: 256

This command defines or displays the distance between the upper and lower limit of the analogue input of the master axis, called the master analogue distance. It enables the value for the analogue distance to be specified for a master axis or enables the value determined using the XR command to be displayed.

CR Calculate initial ratio from analogue range distances

(**C**alculate initial **r**atio from analogue range distances)

"**CR**" calculates the initial transmission ratio from the values determined from the slave and master axis with "**AR**" and "**MM**". The calculation is made using both measured values with "**AR**" (Slave)/ "**MM**" (Master) in the same way as with the SB value, via the BR command.

If the automatic measurement of the analogue distance at the master and at the slave axis is enabled, the "**XM**" command automatically measures the analogue distances at both axes.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.16 Web tension control with analogue input

XRn Execute analogue range distance initialisation

(E**x**ecute analogue **r**ange distance initialisation)

Permitted values for n:
- 0: The slave axis is initialised
- 1: The master axis is initialised

The function: "Initialisation of transmission ratio with analogue distances" can be started manually with “**XR**”. The measurement results are entered in the “**AR**” and “**MM**” parameters.

The command is always triggered on the slave axis. If the master axis is to be initialised, the “**ML**” command must have been executed first.

"Initialisation of transmission ratio with analogue distances" operates as follows:

The motor moves at the velocity “**SS**” in the same direction as the approach of the web tension setpoint value, until an analogue limit value is exceeded; it then stores this position and stops. The motor then moves in the opposite direction until another analogue limit value is exceeded; it then stores this second position and stores the distance between these two values as the value of “**AR**” for slave and “**MM**” for master.

Bits 4 and 5 enable the "Initialisation of transmission ratio with analogue distances", i.e. this function is executed automatically when a position mapping is activated with “**XM**”. The calculated transmission ratio “**AR**” (Slave)/MM(Master) is used as the initial transmission ratio (see also CR command).

3.16 Web tension control with analogue input

AHnn Set analogue input high limit

(Set **a**nalogue input **h**igh limit)

Permitted values for n: ±31Bit value
Factory setting: +2000

This command can be used to set a permitted upper limit for the analogue input. If the value of the analogue input exceeds this value, PMCprimo issues a relevant error message. If Bit 3 of the EW command is set to 1, exceeding the upper limit will be assessed as a “Motor-OFF error”.

ALnn Set analogue input low limit

(Set **a**nalogue input **l**ow limit)

Permitted values for n: ±31Bit value
Factory setting: -2000

This command can be used to set a permitted lower limit for the analogue input. If the value of the analogue input falls below this value, PMCprimo issues a relevant error message. If Bit 3 of the EW command is set to 1, falling below the lower limit will be assessed as a “Motor-OFF error”.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

All numeric values can also be replaced by variables. Variables can be read and modified in PMCprimo via an operator terminal.

All numeric values (identified in this manual with “**n**”, “**nn**” or “**m**”) can be replaced by variables. The variables can be read in and modified via an operator terminal. For example, a position can be specified in a program as a variable rather than a numeric value, and the value of the variable can then be modified by machine operators via the operator terminal. Variables can also be used as trigger variables. As such it is possible to start programs via the operator terminal.

The name of a variable is defined as follows:

- The name of a variable can be up to 20 characters long. Figures from 0 to 9 and letters from A to Z can be used, as well as ‘_’ and ‘**.**’.
- The name of the variable may not contain spaces.
- Upper and lower case letters are equivalent; PMCprimo does not distinguish between them.
- Each variable is preceded by a '**$**' sign to distinguish it from the other commands.

The value can be transferred to a variable in decimal or hexadecimal format. For a hexadecimal format, the figure must be preceded by '0x'.

Example:

PMCprimo user		Comment
`0.2>`	`$SPD=5000<CR>`	With this command, the value of the variable is set to 5000.
`0.2>`	`$SPD=0x1388<CR>`	With this command, the value of the variable is also set to 5000.

Variables can be used instead of the numeric values identified with “**n**”, “**nn**” or “**m**”:

Example:

PMCprimo user		Comment
`0.2>`	`SV$SPD<CR>`	This command is used to assign the value of the variable $SPD to the velocity.
`0.2>`		

3.17 Variable commands

Conversely, parameter values can be also be read into variables:

Example:

PMCprimo user		Comment
0.2>	$SPD=SV<CR>	This command is used to read the value of the velocity into the variable
0.2>		$SPD.

A variable can be defined as a trigger variable, i.e. if the value of this variable is updated, the program assigned to this variable will be run.

Example:

PMCprimo user		Comment
0.1>	$SPD>PROG1<CR>	If the value of the variable $SPD is changed,
0.1>		the program PROG1 is called.

Variables can be used for arithmetic expressions. The user can add with '+', subtract with '**–**', multiply with '*****', divide with '**/**' and implement the modulo function with '**%**'. As variables may only be integer values, the remainder of a division is lost. With the modulo function it is possible to determine the remainder when the left-hand value is divided by the right-hand value. Each bracket that is opened must be closed. An arithmetic expression can be used in exactly the same way as a simple variable.

Example:
Axis 1 must reach a certain start position before the machine starts. The start position is any remainder arising, because in this example the cycle length is not a multiple of the product length.

PMCprimo user		Comment
0.1>	MA(($LEN*20)%$BND)<CR>	The demand position to be reached is the remainder arising from: Product length x 20 divided by the cycle length. $LEN: Product length 600 increments, $BND: Cycle length 5000 increments, giving a demand position of 2000 increments.

Variables should not be used instead of values with a ± sign. The variable "**$A**" contains the value 2 and the user enters "**DR$A-**"; this does not mean "**DR2-**" and PMCprimo registers a syntax error.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

If a command contains more than one numeric value, a variable can be used for each of these values.
Example (setting a transmission ratio via a variable):

PMCprimo user		Comment
1>	$PC=75<CR>	The variable “**$PC**” is set to 75.
1>	SM$PC,100<CR>	The transmission ratio is set to 75:100.
1>		

Variables and arithmetic expressions must be read and calculated as they are executed. As a result, it will take longer to process commands containing variables than it will to process commands in which numeric values are stated.

Variables may also be used in logic operations. These are applied when using the IF command for conditional program processing. A logic operation contains two values which are used with “**==**”, “**!=**”, “**<**”, “**>**”, “**<=**”, “**>=**” to give the result TRUE or FALSE. With arithmetic such as “**&&**” and “||”, corresponding mathematical expressions can result. Each of these expressions must be enclosed in brackets. All brackets that are opened must also be closed.

3.17 Variable commands

The following operations are possible:

Priority	Symbol	Operator	Example	Result
0	( )	Brackets	$a=(5+2)*2	$a=14
1	– + ! ~	Neg. sign Pos. sign Log. negation Bitwise negation	$a=-2 $a=+2 $b=1;$a=!$b $b=255;$a=~$b	$a=-2 $a=2 $a=0 $a=-256
2	* / %	Multiplication Division Remainder	$a=5*3 $a=5/2 $a=5%2	$a=15 $a=2 $a=1
3	+ –	Addition Subtraction	$a=5+3 $a=5-3	$a=8 $a=2
4	a<<n a>>n	Logical left shift: 0 is shifted from the right. The left-hand “**n**” positions are lost, even if “**a**” is signed. Arithmetic right shift: If “**a**” is signed, it will be shifted from the left.	$b=8;$a=$b<<2 $b=20;$a=$b>>2 $a=5	$a=32 $a=5
5	< > <= >= == !=	Less then comparison Greater than comparison Less than or equal to Greater than or equal to Equal to Not equal to		
6	& ^ | && ||	Bitwise AND Bitwise XOR Bitwise OR Logic AND Logic OR	$b=250;$a=$b&15 $b=15;$a=$b^2 $b=15;$a=$b|1 $a=(5==4)&&(5>4) $a=(5==4)¦¦(5>4)	$a=10 $a=13 $a=15 $a=0 $a=1

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

$var=nn Variable assignment

(Variable assignment)

The equals sign '=' is used to assign a particular value, a command, another variable or an arithmetic expression to a variable. The assignment can be used to define a variable.

Example:

PMCprimo user		Comment
0.1>	$SPD=5000<CR>	The value 5000 is assigned to the variable $SPD.
0.1>	CH1;$POS=DP<CR>	The current actual position of axis 1 is read
0.1>		into variable $POS.

NPName Variable number program assignment

(Variable **n**umber **p**rogram assignment)

This command displays the program number of the program 'Name'. The program number is an internal number (address) specified by the compiler; this number can also be used to start a program. NP can be assigned to a variable.

Example:

PMCprimo user		Comment
0.1>	$VAR=NP PRG1<CR>	The program number from 'PRG1' is assigned to the variable $VAR.
0.1>	XS $VAR<CR>	Program PRG1 is started.
0.1>		

3.17 Variable commands

NTName **Variable map number assignment**

(Variable map number assignment)

This command displays the table number of the position table 'Name'. The table number is an internal number (address) specified by the compiler; this number can also be used to start a position mapping. NT can be assigned to a variable.

Example:

PMCprimo user		Comment
0.1>	$VAR=NT MAP1<CR>	The table number from 'MAP1' is assigned to the variable $VAR.
0.1>	XM $VAR<CR>	The position mapping is started with MAP1.
0.1>		

WV$var **Wait for write to variable**

(**W**ait for write to **v**ariable)

With this command, PMCprimo waits until the stated variable is assigned a value before executing the commands that follow. A status variable or error variable may not be used for this command. However, it is possible to use the pre-defined map variable "$MREADY". This variable uses the internal motion generator.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

$var>[PrgName][,0|1] **Define trigger variable**

(Define trigger variable)

The '>' sign is used to define a variable as a trigger variable. The defined program is run each time the variable is activated via the operator terminal or when the variable value changes. Programs that are assigned to a trigger variable are not permitted to write to the variables themselves, as this would create a continuous loop (see also Disable/inhibit trigger variable from Page 3-219).

If ", 1" is stated after the program name, the trigger program is only executed when a value changes. If ", 0" is stated, or no details are stated, the program will be executed each time the variable is activated.

Example:

PMCprimo user		Comment
1>	$SPD>PRG1<CR>	The program PRG1 is executed each time the variable is activated.
1>		

If a command line is not stated, the definition as trigger variable is invalidated.

All trigger variables are disabled when the control system is switched on. They are only enabled through the “**EV**” command.

3.17 Variable commands

IA $var[n] Define variable array

(Define variable array)

Permitted values for n: 1 to 255

This command defines a variable array. Each individual element can be stored in MRAM. Each element is a fully-fledged variable.

Example:

PMCprimo user	Comment
`1.1>  IA $SPD[5]<CR>`	Variable array with 5 elements is defined.
`1.1>  $SPD[3]=13`	Value is assigned to the third element.
`1.1> $SPD`	Value query, variable array
`1.1>`	
`1.1> $SPD[0]=0  (0x00000000)`	
`1.1> $SPD[1]=0  (0x00000000)`	
`1.1> $SPD[2]=0  (0x00000000)`	
`1.1> $SPD[3]=13  (0x0000000D)`	
`1.1> $SPD[4]=0  (0x00000000)`	
`1.1> $SPD[3] > $H10`	Element 3 of variable array $SPD is stored in MRAM.
`1.1>`	

$var>$H1...1000 Store variable in MRAM

(Store variable in MRAM)

Variables in PMCprimo can be stored in non-volatile memory (MRAM). A total of 1000 variables can be stored. This is also possible with bus variables $B1...$B100.

Example: `$VAR > $H12`

The variable '$VAR' is linked to storage space '$H12' on the MRAM. The “**LV**” command displays this link.

On power-up, a check sum is used to check whether any data on the MRAM has been lost. If this is the case, the variables will be initialised with the last values to be stored via the “**SP**” command.
Storage in the MRAM is cancelled with “$VAR >”.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

VX List trigger variables

(List trigger **v**ariables)

This command can be used to list all trigger variable definitions. The first letter of a displayed line has the following meaning:

E: The trigger variable is enabled
M: The trigger variable is masked (MV command)
B: The trigger variable is inhibited (BV command)

Example:

PMCprimo user	Comment
`0.1>  VX<CR>`	All trigger variable definitions are listed.
`M - $SPD>PRG1`	Variable is masked.
`E - $TR>START_PRG`	Variable is enabled.
`B - $FRG>PRG2`	Variable is inhibited.
`0.1>`	

MV[$var] Mask trigger variables

(Mask trigger variables)

This command can be used to mask a trigger variable. In this case, "$var" represents the name of the variable that is to be masked. If "$var" is not stated, PMCprimo masks all trigger variables.

If a value is assigned to a masked trigger variable, the defined program will be executed once it is enabled with the “**EV**” command.

The “**VX**” command provides information as to whether a trigger variable is masked, inhibited or enabled.

BV[$var] Inhibit trigger variables

(Inhibit trigger variables)

This command can be used to inhibit a trigger variable. In this case, "$var" represents the name of the variable that is to be inhibited. If "$var" is not stated, PMCprimo inhibits all trigger variables.

If a value is assigned to an inhibited trigger variable, the defined program will **not** be executed once it is enabled with the “**EV**” command.

The “**VX**” command provides information as to whether a trigger variable is masked, inhibited or enabled.

3.17 Variable commands

EV [$var]Enable trigger variables

(Enable trigger variables)

This command can be used to enable a trigger variable. In this case, "$var" represents the name of the variable that is to be enabled. If "$var" is not stated, PMCprimo enables all trigger variables.

LV List variable value

(**L**ist **v**ariable value)

This command is used to list all the user-defined variables and their current values. If a variable name is entered, its value will be listed.

Example 1:

PMCprimo user	Comment
`0.1>  $SPD<CR>`	User request
`$SPD=5000  (0x00001388)`	The value of the variable is 5000.
`0.1>`	

Example 2:

PMCprimo user	Comment
`0.1>  LV<CR>`	User request
`$BCD=12`	The value of the variable BCD is 12
`$SPD=5000`	The value of the variable SPD is 5000
`$TR=75`	The value of the variable TR is 75
`0.1>`	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

LB List bus variable value

(**L**ist **b**usvariable value)

This command can be used to list the pre-defined bus variables. There are 108 bus variables in total, with the following syntax:

$Bn n: from 1 to 108

These bus variables are used as an interface to external devices, such as an operator terminal.

Variables $B1 to $B100 can be defined as trigger variables and should be used as "normal" variables. To query the value of an individual variable, all you need to do is state its name.

The variable values $B101 to $B108 are mapped to the virtual input banks 0.101 to 0.108. By assigning a value, e.g. $B101=0xff, all the inputs of input bank 0.101 are set. These inputs have the same functionality as the CAN inputs. Corresponding outputs, such as those found on classic virtual inputs, are not available, as they are switched via the bus variable.

The values for $B101 to $B108 cannot be saved. Neither can they be defined as trigger variables. The address range of Modbus, CANbus and Profibus is increased accordingly. The variables cannot be used directly as command parameters (variables cannot be global).

Example:

PMCprimo user		Comment
`0.1>`	`$B12<CR>`	User request
`$B12=8`	`(0x00000008)`	The value of the bus variable is 8.
`0.1>`		

LP List map variable value

(**L**ist ma**p**variable value)

This command is used to list the variables defined for the internal motion generator (see "***PMCprimo*** Motion Generator **PMotion** User Manual").

The variables $MMSB and $MSSB are also listed.

3.17 Variable commands

$Fn Set error variable

(Set error variable)

Permitted values for n: 1 to 60 (can be set via CD command)

There is a pre-defined error variable for each axis. Its name is made up as follows:

$FAxis number

The error variable stores the error number of the last error that occurred. An error variable can be used as a trigger variable. A value can be assigned to an error variable. Possible values for the error variables can be found in Chapter 4.2, on Page 4-230.

The last error number from the host, CAN, map generator and compiler is stored in the error variable $F0. The error messages “Cannot start program because it is already running” and “Program is not available” do not generate an error number for $F0, otherwise continuous loops may occur when $F0 is defined as a trigger variable.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.17 Variable commands

$Sn Set motor status variable

(Set motor status variable)

Permitted values for n: 1 to 60 (can be set via CD command)

There is a pre-defined motor status variable for each axis. Its name is made up as follows:

$SAxis number

The status variable contains the current motor status. An status variable can be used as a trigger variable. A value can be assigned to a status variable. The possible values for the status variables can be found in Chapter 4.3, Page 4-253.

$Wn Set status variable

(Set status variable)

Permitted values for n: 1 to 60 (can be set via CD command)

There is a pre-defined wait status variable for each axis. Its name is made up as follows:

$WAxis number

The status variable contains the current wait status. An status variable can be used as a trigger variable. A value can be assigned to a status variable. The possible values for the status variables can be found in Chapter 4.3, Page 4-253.
The size of this variable is just 9 Bits. Each bit has a different meaning:

Bit 2: WA active
Bit 3: WR active
Bit 4: WF active
Bit 5: WB active
Bit 6: WC active

3.18 Commands for CANopen network

In a CANopen network, data is sent via the following objects:

SDO Service Data Object. Needed to configure the bus node. Transmission is confirmed.

PDO Process Data Object. Needed for real-time data exchange. Transmission is unconfirmed.

From the PMCprimo side there are commands for reading and writing SDOs:

QSt,i,s,c,n Send SDO

(Send SDO)

Parameter t:	CAN nodes
Parameter i:	Index
Parameter s:	Subindex
Parameter c:	No. of Bytes
Parameter n:	Value

With this command, PMCprimo can access the object directory of a CAN device via SDO and so can write set values or configure the device, for example. Please refer to the manual of the respective network subscriber for details of how to configure the parameters.

QRt,i,s,c,n Read SDO

(Read SDO)

Parameter t:	CAN nodes
Parameter i:	Index
Parameter s:	Subindex
Parameter c:	No. of Bytes

With this command, PMCprimo can access the object directory of a CAN device via SDO and so can read actual values, for example. Please refer to the manual of the respective network subscriber for details of how to configure the parameters. The value that is read can be assigned to a variable.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.18 Commands for CANopen network

PDt, x **Position from drive**

(**p**osition from **d**rive)

Permitted values for t: 0 to 150 (1-50 CANopen, 51-150 SafetyNet)
Permitted values for x: 0 to number of encoder inputs

This command defines the connection between a servo amplifier and an axis channel.

The address range of 1 to 50 is reserved for CANopen.

The address range of 51 to 150 is reserved for SafetyNet.

The value 0 cancels the definition and disconnects the connection via CAN.

The "**FS**" command must be set so that position values are transferred via CANopen / SafetyNet (FS19 – FS27).

The cycle time and baud rate for the CAN bus can be set via the "**CD**" command. Many different servo amplifiers can be networked with the PMCprimo via CAN, depending on the setting.

Cycle time in ms	Max. no. servo amplifiers 500 Kbit per CANopen	Max. no. servo amplifiers 1 Mbit per CANopen
1	2	4
2	4	8
4	8	16
8	16	32

With 2 CAN networks, therefore, it is possible to network up to a maximum of 49 servo amplifiers.

CNnn [,X1,X2] **CAN trace**

(**CAN** Trace)

Permitted values for n: 0 to 3
Permitted values for X1: 0 to 4
Permitted values for X2: 0 to 127
Factory setting: 0

This command can be used to record a CAN trace.

3.18 Commands for CANopen network

nn = 0: The CAN trace function is disabled, recorded CAN traces are deleted

nn = 1: Starts recording the CAN trace.

With X1 there is an option to select which CAN network is to be recorded.
X1= 0: All networks are recorded
X1=1: CAN 1 is recorded (default)
X1=2: CAN 2 is recorded
With X2 there is an option to set a filter on a CAN address.
X2=0: No filter active (default)
X2=1..127: Passthrough filter for the stated node number

nn = 2: Displays the recorded CAN trace on PTerm.

nn = 3: Recorded CAN traces are deleted and recording is restarted.

nn = 4: Recording is started with ring memory. This enables continuous recording.

nn = 5: This can be used to stop the continuous recording that was started with CN4.

If a CAN trace is to be made of the bus scan, **CN**1 must be triggered and must be stored afterwards with **SP**. When PMCprimo is restarted, the bus scan is recorded and can then be displayed directly with **CN**2 after booting.

Example 1:

PMCprimo user	Comment
0.1: cn0	Delete the Can trace
0.1: cn1,1,50;qr50,0x1000,0,4	Start CAN trace on main network, filter for
0.1:	CAN address 50 and SDO on address 50
0.1:	
0.1 QR=131474 20192 (hex)	Response from SDO
0.1: cn2	Display CAN trace
0.1:	
Time 16200392 CanNo 1 CobID 0632 Length 8 Data 40 00 10 00 00 00 00 00	
Time 16200393 CanNo 1 CobID 05b2 Length 8 Data 43 00 10 00 92 01 02 00	
Time 16200496 CanNo 1 CobID 0732 Length 1 Data 00	
Time 16200496 CanNo 1 CobID 0732 Length 1 Data 85	
Time 16200746 CanNo 1 CobID 0732 Length 1 Data 00	
Time 16200746 CanNo 1 CobID 0732 Length 1 Data 05	
Time 16200996 CanNo 1 CobID 0732 Length 1 Data 00	
Time 16200996 CanNo 1 CobID 0732 Length 1 Data 85	

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

3.19 Commands for the safety card

Commands for communication with the safety card

SD Display IO state safety card

(Display IO-**S**tate Safetycar**d**)

This command is used to display the safety card's IO state register.

Byte 1 (inputs)	Byte 2 (outputs)
1 = SS1 Activate	1 = READY
2 = I0 (SS2 Activate)	2 = STO Acknowledge
3 = I1 (SOS Activate)	3 = O0 (SOS Acknowledge)
4 = I2 (SLS Activate)	4 = O1 (SDI Acknowledge)
5 = I3 (SSR Activate)	5 = O2 (SRA Acknowledge)
6 = I4 (SDI left Activate)	6 = O3 (SBT Acknowledge)
7 = I5 (SDI right Acitvate)	7 = SBC
8 = I6 (SBT Activate)	8 = SIL3 STO

Byte 3	Byte 4
1 = Reserved	1 = Reserved
2 = Reserved	2 = Reserved
3 = Reserved	3 = Reserved
4 = Reserved	4 = Reserved
5 = Reserved	5 = Reserved
6 = Reserved	6 = SS2 is 0 (drive-activated) / 1 (controller-activated)
7 = Reserved	7 = SS1 is 0 (drive-activated) / 1 (controller-activated)
8 = Reserved	8 = SS1 internal

3.19 Commands for the safety card

CSnn [,X1,X2] Communication safety card

(**C**ommunication **S**afetyCard)

This command can be used to communicate with the safety card:

nn = 1: Displays a list of all active safety cards.
nn = 2: Download the configuration data to the safety card with serial number X1.
nn = 3: Display the status information of the safety card with serial number X1
nn = 4: Read the error stack of the safety card with serial number X1. All errors are displayed when X2 = 0; when X2 = 1, only the active errors are displayed.
nn = 5: Upload the configuration data from the safety card with serial number X1.

Example 1 (CS 3):

PMCprimo user					Comment
0.1: CS 3 , 680000 100001					
680000 100001 0.1	00000203	00000000	2001817F	1843	
(1)	(2)	(3)	(4)	(5)	

(1) Serial number of safety card
(2) State register
(3) Error register
(4) I/O state register
(5) Current velocity

With CS commands, a distinction is made between DIS1 and DIS2. See safety card documentation.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4 Status and error messages

4.1 Status messages

Overview of all the status messages that PMCprimo shows to the user on screen via a Telnet connection:

>	Position control loop closed and PMCprimo ready for the next travel command.
:	Motor switched off – Position control loop open.
A	Axis executes an alignment movement for starting a position mapping (XM command). Once this alignment movement has been completed, message X appears for position mapping active.
C	Axis executes the coupling process or (MW command Bit 0 = 1) decoupling process (STx) for starting a position mapping (XM). Once this process movement has been completed, message X appears for position mapping active.
I	Initialisation running.
M	Axis executes a positioning with a trapezoidal velocity profile.
S	Axis executes a stop command.
V	Execution of infinite positioning or velocity control.
W	Axis is in standby.
X	Position mapping is active on the axis and alignment movement or coupling process has been completed.
N	Linear interpolation is active.
nnnE	Program entry (see “**ES**” command, Page 3-59).
nnnM	Map definition (see “**EM**” command, Page 3-80)

4.2 Error messages

Overview of all the error messages that PMCprimo shows to the user on a display unit:

Code	Error
0	No error
1	Cannot start program because the max. number of running processes has been reached!
2	This command is not yet implemented on the control system!
3	Cannot execute the “**MA**” command because the motor is off!
4	The target position is outside the defined software end limits!
5	Software end limit has been passed in the plus direction
6	Software end limit has been passed in the minus direction!
7	Cannot execute the “**MA**” command because the motor is already running!
8	Cannot execute the “**MR**” command because the motor is off!
9	Cannot execute the “**MR**” command because the motor is already running!
10	Cannot execute the “**VC**” command because the motor is off!
11	Cannot execute the “**VC**” command because the motor is already running!
12	Attempt to divide by zero!
13	Switch to a sequential command sequence before calling a program!
14	Data stack overflow!
15	Map generator is already running
18	Map generator error: Variable <$MNAME> is not defined
19	Map generator error: Variable <$MNPT> is not defined
20	Variable <…> is already being used for <WV>
21	Input is not defined as a counter input!
22	Input is not defined as an enable input!
23	Not enough RAM!
24	Position error limit has been exceeded!
25	Input cannot be defined as a 'reset input'!

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

Code	Error
26	No trigger variable is defined!
27	Cannot start the program <....> because it is already running!
28	Could not achieve the target position!
29	No rotary encoder signals received!
30	Error reading the rotary encoder
31	Cannot execute the “**XM**” command because the motor is off!
34	Too much time was needed to generate the set value!
35	Cannot execute program <...> because it is does not exist!
36	“**ML**” must be executed before “**BR**”
37	Reference error limit has been exceeded!
38	Could not correct the reference error in one cycle!
39	The monitoring time for the reference signal has elapsed!
40	The reference error has been reduced to the maximum value!
41	Program <...> is currently inactive
42	Velocity mapping only possible with linear map
47	The input number may not be greater than 8!
48	This input group number is not defined!
49	The output number may not be greater than 8!
50	The output group does not exist!
52	The output has already been defined as an 'error output'!
53	The output has already been defined as an 'electronic cam'!
54	The output has already been defined for 'Pulse at cycle limit'!
55	Undefined error message!
56	The limit switch has been activated and the motor switched off!
57	The input is already defined!
58	The input is not defined as a 'limit switch'!
59	At least 2 pairs of values must be defined for map generation
60	Map generator error: Variable $MSnnn has not been defined
61	Map generator error: Invalid value for $MXnn

4.2 Error messages

Code	Error
62	Map generator error: Invalid value for $MYnn
63	Target position is outside the software end limit
64	Variable for map generation is not defined
66	The parameter value <...> is not permitted for the <...> command!
67	Cannot execute MI. command because axis is in mapping
68	Map generation error: Variable $MM%Z is less than its preceding value!
69	Input cannot be defined as an 'input function'!
70	Input cannot be defined as an 'enable input'!
71	Input cannot be defined as a 'counter input'!
72	Master and slave axis must be different!
73	The axis <...> does not exist (<...> command)!
74	Monitoring time for master positions has elapsed
75	Cannot execute QR command (CAN-Bus not ok?)
76	Cannot execute <PC> because analogue auxiliary output is defined!
77	The output has already been defined as an analogue auxiliary output!
78	An encoder snapshot has already been defined for this axis!
79	Input is not a 'fast input'!
80	The 'fast input' is already defined
81	No reference signal has been defined
82	“**IB**” command is not possible because axis is virtual and motor is off
83	Cannot execute “**LW**” command because assignment as slave has already been made via “**ML**”
84	The output is defined and so cannot be switched manually
85	The output has already been defined as a reference output
86	The output has already been defined as an error output for the analogue input limit
87	The output is not defined
88	The output has already been defined as a reference error correction
89	The encoder zero track is already defined for the axis

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

Code	Error
90	An input is defined as the encoder snapshot for the axis
91	An input is defined as the reference input for the axis
92	The input is already being used for another axis
93	Fast input is not defined as an encoder snapshot!
94	Fast input is not defined as a reference input
95	The output has already been defined for 'Pulsed output during reference signal'
96	Incrementing error: Upper limit of 2*“**SV**" has been exceeded
97	Incrementing error: Creep speed may not be greater than “**SV**”
98	Incrementing may not change the direction of velocity
99	Cannot execute the “**XR**” command because the motor is off!
100	Cannot execute the “**XR**” command because the motor is already running!
101	Map generator error: Variable $MFnn has not been defined
102	Map generator error: Variable $MWnn has not been defined
103	Map generator error: Variable $MAnn has not been defined
104	Map generator error: Variable $MZnn has not been defined
105	Map generator error: Variable $MXnn has not been defined
106	Map generator error: Variable $MYnn has not been defined
107	Map generator error: Variable $MXnn > $MYnn
108	Map generator error: Variable $MBnn has not been defined
109	Map generator error: Variable $MCnn has not been defined
110	Map generator error: Segment No.nn: Percentage values must add up to 100%
111	Map generator is not enabled
114	The output has already been defined as “**<JF>**”
115	“**<XR>**” / “**<XM>**” command was cancelled
116	BUS variable is not defined
117	Value for analogue input is below the lower limit value

4.2 Error messages

Code	Error
118	Upper limit value for analogue input has been exceeded
119	"**ML**" must be executed before "**XV**"
120	The input is defined for an encoder snapshot
121	The input is defined as a reference input
122	Cannot activate position control loop because initialisation is active
123	Cannot define analogue auxiliary output because position control is active
124	Cannot switch to real mode because an analogue auxiliary output is defined.
125	Transmission ratio can only be incremented if the position table is active
126	Program <....> does not exist
127	Invalid output number
128	Invalid output group number
129	Value range for shift register with "**TC**" command is between 1 and 32
130	FC command only possible when position control loop is switched off
131	VM command only possible when position control loop is switched off
132	When the master is moving, engagement cannot take place without software coupling
133	Cannot execute "**VC**" command because axis is in mapping!
134	Wait loop for absolute position is already active on the axis
135	Wait loop for relative position is already active on the axis
136	Wait loop for reference input is already active on the axis
137	Wait loop for cycle limit is already active on the axis
138	Wait loop for overflow counter is already active on the axis
139	Axis can only be a virtual axis
140	Position table "**<....>**" does not exist
141	Master value does not exist in the position table
142	Axis does not exist
143	Variable is not defined
144	Invalid network subscriber number when naming inputs

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

Code	Error
145	No program is defined for the input
146	A fast input is already defined for the axis
149	No more than 8 master axes can be run via the CAN bus
150	Cannot execute “**UL**” command because position mapping is active
151	“**ML**” must be executed before “**XM**”
152	External master value is not permitted with “**XR**”
153	Caution. Data will be deleted because the check sum calculation is invalid
154	Monitoring is already running
155	The following baud rates can be set: \r\n9600\r\n14400\r\n19200\r\n38400\r\n57600\r\n115200
156	No recording value has been selected
157	If monitoring is running, **“TW”** may not be modified
158	Channels >4 do not exist
159	Invalid data type for “**TW**” command
160	No programs are defined
162	Mode <...> cannot be selected for “**TC**” command
163	Invalid counter value <...> for “**TC**” command
165	Caution!!! Error during software update
171	“**NL**” must be executed for software differential
172	Controller error: Heat sink temperature > 80°C
173	Controller error: Overvoltage on intermediate circuit
174	Controller error: Open circuit, short circuit in feedback (Resolver, Hiperface)
175	Controller error: Undervoltage on intermediate circuit
176	Controller error: Motor limit temperature exceeded
177	Controller error: Problem with internal supply voltage
178	Controller error: Maximum velocity exceeded
179	Controller error: EEprom check sum error
180	Controller error: Flash-Eprom check sum error

4.2 Error messages

Code	Error
181	Controller error: Open circuit or short circuit on brake
182	Controller error: Motor phase missing
183	Controller error: Device's internal temperature too high
184	Controller error: Power stage defective
185	Controller error: I2t limit exceeded
186	Controller error: 2 or 3 phases missing in the supply voltage
187	Controller error: A/D converter faulty
188	Controller error: Ballast resistor defective or faulty setting
189	Controller error: Phase missing in the supply voltage
190	Controller error: System software not in order
191	Controller warning: I2t threshold exceeded
192	Controller warning: Preset ballast output has been reached
193	Controller warning: Watchdog for expansion card (PMCprimo) has been activated
194	Controller warning: Supply voltage phase missing
195	Controller warning: Hiperface: Basic setting loaded from motor
196	Controller warning: Expansion card (PMCprimo) is not operating correctly
197	Controller warning: Enable signal not connected
198	CAN bus error: Initialisation: Safety cables are not online
199	CAN bus error: When writing PDO data: CAN not in operational mode
200	CAN bus error: When writing to the pipe
201	CAN bus error: The safety cable was connected by a subscriber
202	CAN bus error: The 12V supply voltage has been switched off
203	CAN bus error: Overflow on receive buffer: Data has been lost
204	CAN bus error: Overflow on send buffer: Data will be lost
205	CAN bus error: The CAN controller has been switched off due to too many errors
206	CAN bus error: The receive buffer in the CAN controller has overflowed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

Code	Error
207	CAN bus error: An error occured during transmission
208	CAN bus error: The CAN module watchdog has been activated
209	CAN bus error: There was an error in the SDO transmission: Time exceeded
210	CAN bus error: An error occurred during the SDO transmission, e.g. incorrect no. of Bytes
211	CAN bus error: Error message, this message is not implemented
212	CAN bus error: Too many CAN errors: CAN bus will be switched off
213	CAN bus inactive!
214	CAN node “**nn**”: Firmware version does not match the host
215	Error: Servo controller cannot be switched on via CAN
216	Error: Servo controller cannot be switched off via CAN
217	PD command only possible from axis “**nn**”!
218	CAN bus error: Node guarding error, node “**nn**”!
219	Caution. Data loss in battery-buffered memory
220	Battery-buffered memory is not available within the system
221	Position table “**<...>**” has not been sent to axis (TM command)
223	Mapping is not active
224	Servo controller error via CAN
225	Caution: Bit 1 of “**FW**” has been set to 1, as RJ>SB
226	Variable could not be assigned to drive command!
227	No data present in Flash!
228	Insufficient Flash memory!
229	Error occurred when saving!
230	No more reference positions can be buffered!
231	The buffered reference positions will be deleted due to the change in the direction of rotation!
232	The following baud rates can be set: \r\n9600\r\n19200(XOn/XOff)\r\n38400(XOn/XOff)
233	The output has already been defined as a 'Velocity-dependent output'!

4.2 Error messages

Code	Error
234	Too many PLC tasks active
235	No valid data from TR
236	Output definition not possible on CAN-IO module
237	TD<...>,<...> is not defined!
238	The output has already been defined for monitoring the position error!
239	Cannot execute the "**ID**" command because the motor is not in 'PC' status
240	The parameter value "**<%ld>**" is not permitted for the command <...> if Bit 4 of "**RW**" is set
241	The parameter value "**<%ld>**" is not permitted for the command <...> if Bit 4 of "**RW**" is **not** set
242	The parameter value "**<%ld>**" is not permitted for the command <...> if Bit 6 of "**RW**" is set
243	The parameter value "**<%ld>**" is not permitted for the command <...> if Bit 6 of "**RW**" is **not** set
244	Value of RC<1000 is not permitted because Bit 7 of RW was set to 0
245	Value of RV>200 is not permitted because Bit 5 of RW was set to 0
246	Bit 5 of RW is automatically set to 1 because Bit 7 was set to 1.
247	Internal PLC error
248	Could not execute drive command because a node is not a drive
249	Controller error: Commutation error
250	Variable "**$<...>**" is not defined
251	Network subscriber number "**<...>**" is not assigned ("**<...>**" command)
252	Network subscriber "**<...>**" is already being accessed ("**<...>**" command)
253	No CAN device found
254	Cannot execute drive command due to a buffer overflow
255	Warning: Discharge battery for SRAM
256	Controller error: Issue enable before AS option
257	Controller error: Earth fault
258	Controller warning about CAN
259	Available CompactFlash memory "**<...>**" Bytes
260	Cannot execute the "**XM**" command because the motor is already running
261	QA0 command: CAN addr. can only be executed on the host
262	The analogue output "**<...>**" does not exist! (<%c%c> command)
263	Reserved for soft PLC

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

Code	Error
264	Reserved for soft PLC
265	Reserved for soft PLC
266	Reserved for soft PLC
267	Reserved for soft PLC
268	Reserved for soft PLC
269	Reserved for soft PLC
270	Reserved for soft PLC
271	Reserved for soft PLC
272	Reserved for soft PLC
273	PD command not possible! Incorrect PMCtendo DD4 version (< 4.94)
274	Node <...>, drive command incomplete!
275	Message from soft PLC: CAN device is not a DD4
276	SoftSPS is using battery-buffered memory.
277	The maximum possible position (SB*SM) exceeds the value range of %ld (<%c%c> command)
278	Cannot modify the pre-defined position table 'LIN‘AR'.
279	The interpolator for the CNC application is active, so cannot execute the %c<%c%c> command.
280	Note: The ZC command has no function on absolute encoders!
281	Error: Number of Ethercat systems has changed! Axes will be stopped
282	CAN error: %s
283	“**ML**” must be executed before “**XI**”
284	XI command: Master position was not found
285	Slave value does not exist in the position table
286	Output is reserved for PLC
287	Cannot execute PC command because the controller is not ready for operation
288	Module on slot %ld of node %ld may not be moved up
289	Module or module configuration on slot %ld of node %ld is not supported
290	Node no. %ld: Too many digital input modules used or more than 4 TPDOs required
291	Node no. %ld: Too many digital output modules used or more than 4 RPDOs required
292	Node no. %ld: Too many analogue input modules used, 8 analogue inputs are supported

4.2 Error messages

293	Node no. %ld: Too many analogue output modules used, 8 analogue outputs are supported
294	Node no. %ld: Digital CAN-IO input modules (ES DI) masked
295	Node no. %ld: Digital CAN-IO output modules (ES DO) masked
296	Node no. %ld: Analogue CAN-IO input modules (AI U or AI I) masked
297	Node no. %ld: Analogue CAN-IO output modules (AO U or AO I) masked
298	Node no. %ld: Safe digital CAN-IO input and output modules (FS DI or FS DO) masked
299	Warning: The device with node number %i does not support node guarding and is not monitored
300	The controller cannot be enabled
301	The controller cannot be disabled
302	Error in version description
303	Motion generator error, status% i
304	Safety card error - ERR01: Serial number not found
305	Safety card error - ERR03: CRC check sum invalid
306	Safety card error - ERR07: Timeout
307	Safety card error - ERR04: Incorrect configuration data
308	No safety card detected or position control not active
309	Bus load measurement error
310	Safety card error - Safe Stop 2 (SS2): Monitoring of standstill position failed
311	Safety card error - Safe Operating Stop (SOS): Monitoring of standstill position failed
312	Safety card error - Safely Limited Speed (SLS): Monitoring of safely limited speed failed
313	Safety card error - Safe Speed Range (SSR): Monitoring of speed range failed
314	Safety card error - Safe Direction (SDI): Monitoring of direction of rotation failed
315	Safety card error - Safe Brake Test (SBT): Monitoring of standstill position (motor brake) failed
316	Safety card error - Safe Brake Test (SBT): Monitoring of standstill position (external brake) failed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

317	Safety card error - Safe Brake Test (SBT): Monitoring 'Motor current ON' (motor brake) failed
318	Safety card error - Safe Brake Test (SBT): Monitoring 'Motor current OFF' (motor brake) failed
319	Safety card error - Safe Brake Test (SBT): Monitoring 'Motor current ON' (external brake) failed
320	Safety card error - Safe Brake Test (SBT): Monitoring 'Motor current OFF' (external brake) failed
321	Safety card error - Safe Brake Test (SBT): BC function not configured
322	Safety card error - Safe Brake Test (SBT): Test period elapsed
323	Safety card error - Digital input test (input 'SS1 Activate') failed
324	Safety card error - Digital input test (input 'SS2 Activate') failed
325	Safety card error - Digital input test (input 'SOS Activate') failed
326	Safety card error - Digital input test (input 'SLS Activate') failed
327	Safety card error - Digital input test (input 'SSR Activate') failed
328	Safety card error - Digital input test (input 'SDI Neg Activate') failed
329	Safety card error - Digital input test (input 'SDI Pos Activate') failed
330	Safety card error - Digital input test (input 'SBT Activate') failed
331	Safety card error - Digital output test (output 'STO Acknowledge') failed
332	Safety card error - Digital output test (output 'SOS Acknowledge') failed
333	Safety card error - Digital output test (output 'SDI Acknowledge') failed
334	Safety card error - Digital output test (output 'SRA Acknowledge') failed
335	Safety card error - Digital output test (output 'SBT Acknowledge') failed
336	Safety card error - Digital output test (output 'Ready') failed
337	Safety card error - Digital output test (output 'STO (internal STO Enable)) failed

4.2 Error messages

338	Safety card error - Digital output test (output 'SIL3 STO') failed
339	Safety card error – Voltage monitoring test (reference voltage - Safe Clock) failed
340	Safety card error – Voltage monitoring test (supply voltage digital outputs – charge pump [start-up test]) failed
341	Safety card error – Voltage monitoring test (supply voltage digital outputs – charge pump [cycl. run test]) failed
342	Safety card error – Voltage monitoring test (supply voltage digital outputs – charge pump [off test]) failed
343	Safety card error – Voltage monitoring test (supply voltage digital outputs – SafeClock [off test]) failed
344	Safety card error - Watchdog test (Watchdog inactive [start-up test]) failed
345	Safety card error - Watchdog test (Watchdog triggered too early [start-up test]) failed
346	Safety card error - Watchdog test (Watchdog did not trigger [start-up test]) failed
347	Safety card error - Watchdog test (Watchdog inactive [discharge test]) failed
348	Safety card error - Watchdog test (Watchdog did not trigger [discharge test]) failed
349	Safety card error – Memory test (Flash/RAM); memory test error
350	Safety card error - Timing error (error waiting for fast interrupt)
351	Safety card error - Timing error (fast interrupt was called, although it is already running)
352	Safety card error - Timing error (1 ms interrupt was called, although it is already running)
353	Safety card error - Timing error (1 ms interrupt was called too early)
354	Safety card error - Timing error (1 ms interrupt was called too late)
355	Safety card error – Feasibility of position / encoder signal (encoder signal error (motor feedback)) failed
356	Safety card error – Feasibility of position / encoder signal (internal position servo controller (DPR-SSI)) failed
357	Safety card error – Feasibility of parameters (invalid number of motor poles) failed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

358	Safety card error – Feasibility of parameters (invalid motor type) failed
359	Safety card error – Feasibility of configuration data / inputs (configuration data / input states do not match) failed
360	Safety card error – Channel A/B data alignment (state of input 'SS1 Activate') failed
361	Safety card error – Channel A-B data alignment (state of input 'SS2 Activate') failed
362	Safety card error – Channel A-B data alignment (state of input 'SOS Activate') failed
363	Safety card error – Channel A-B data alignment (state of input 'SLS Activate') failed
364	Safety card error – Channel A-B data alignment (state of input 'SSR Activate') failed
365	Safety card error – Channel A/B data alignment (state of input 'SDI Neg Activate') failed
366	Safety card error – Channel A/B data alignment (state of input 'SDI Pos Activate') failed
367	Safety card error – Channel A-B data alignment (state of input 'SBT Activate') failed
368	Safety card error – Channel A-B data alignment (1ms timer) failed
369	Safety card error – Channel A-B data alignment (operating state) failed
370	Safety card error – Flash memory error: Error writing to Flash
371	Safety card error – Flash memory error: Error when deleting a sector
372	Safety card error - Synchronisation channel A-B (Power–n – synchronisation point 1) failed
373	Safety card error - Synchronisation channel A-B (Power–n – synchronisation point 2) failed
374	Safety card error - Synchronisation channel A-B (Start–p – synchronisation point 1) failed
375	Safety card error - Synchronisation channel A-B (Start–p – synchronisation point 2) failed
376	Safety card error - Synchronisation channel A-B (Start–p – synchronisation point 3) failed
377	Safety card error - Synchronisation channel A-B (R–n – synchronisation point 1) failed

4.2 Error messages

378	Safety card error - Synchronisation channel A-B (self test – synchronisation point 1) failed
379	Safety card error - Synchronisation channel A-B (channel B must wait for channel A due to saving the error stack) failed
380	Safety card error - Synchronisation channel A-B (synchronisation download Config-data receipt) failed
381	Safety card error - Watchdog was triggered
382	Safety card error – Checking of data/parameters (checking of configuration parameters) failed
383	Safety card error – Checking of data/parameters (invalid hardware version number on firmware update) failed
384	Safety card error – Checking of data/parameters (invalid hardware version number with CS 0 command) failed
385	Safety card error – Checking of data/parameters (configuration data download: too much data) failed
386	Safety card error – Data transmission timeout when downloading configuration data:
387	Safety card error – Data transmission timeout when uploading configuration data:
388	Safety card error – Data transmission timeout when updating firmware
389	Safety card error – CRC calculation error when downloading configuration data:
390	Safety card error – CRC calculation error when loading the configuration data from the Flash
391	Safety card error – CRC calculation error when updating firmware
392	Safety card error – Checking of device data (checking serial number when downloading configuration data) failed
393	Safety card error – Checking of device data (checking device version when downloading configuration data) failed
394	Safety card error – Checking of device data (checking device type when downloading configuration data) failed
395	Safety card error – Checking of device data (checking device version on run-up) failed
396	Safety card error – Checking of device data (device has no serial number) failed
397	Safety card error – Checking of device data (checking safety version (servo amplifier firmware) on run-up) failed
398	Safety card error – Checking of device data (checking serial number when loading configuration data) failed
399	Safety card error – Checking of device data (checking device version when loading configuration data) failed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

400	Safety card error – Checking of device data (checking device type when loading configuration data) failed
401	Safety card error – Checking of device data (checking hardware version during firmware update) failed
402	Safety card error – Checking of device data (checking hardware version when reading the serial number (CS 0)) failed
403	Safety card error – System software error: Invalid interrupt
404	Safety card error – System software error: Invalid Case statement
405	Safety card error – Unknown error number
406	Safety card error – Safe Stop 1 / Emergency Stop is configured as drive-activated on node no. %ld
407	Safety card error – Safe Stop 2 is configured as drive-activated on node no. %ld
408	The output is already defined as 'alignment run'
409	Safety card error – Feasibility of position / current (zero crossing of current signal not recognised) failed
410	Safety card error – Feasibility of position / current (wrong position at zero crossing of current signal)
411	Safety card error – Feasibility of position / encoder signal (speed value (SSI) not feasible) failed
412	Safety card error – Feasibility of position / encoder signal (could not read SSI position) failed
413	Safety card error – Channel A-B data alignment (state of self test) failed
414	Safety card error – Channel A-B data alignment (state of output 'STO Acknowledge') failed
415	Safety card error – Channel A-B data alignment (state of output 'SOS Acknowledge') failed
416	Safety card error – Channel A-B data alignment (state of output 'SDI Acknowledge') failed
417	Safety card error – Channel A-B data alignment (state of output 'SRA Acknowledge') failed
418	Safety card error – Channel A-B data alignment (state of output 'SBT Acknowledge') failed
419	Safety card error – Channel A-B data alignment (state of output 'READY') failed
420	Safety card error – Channel A-B data alignment (state of output "SBC+') failed
421	Safety card error – Channel A-B data alignment (state of output "SBC-') failed

4.2 Error messages

422	Safety card error – Channel A-B data alignment (state of output 'STO SIL 3') failed
423	Safety card error – Channel A-B data alignment (state of output 'STO') failed
424	Safety card error – Channel A-B data alignment (check sum of configuration data) failed
425	Safety card error - Synchronisation channel A-B (synchronisation download Config-data end) failed
426	Safety card error – System software error: CRC test (Flash) – CRC check failed
427	Safety card error – System software error: Memory test (Flash/Ram) – Memory corrupt
428	Safety card error – System software error: RAM data test
429	Safety card error – System software error: Checking of parameters
430	Safety card error – System software error: Self test
431	Safety card error – System software error: Expiry monitoring
432	Safety card error - Digital output test (dual-pole output 'SBC+/SBC-': monitoring error) failed
433	Safety card error - Digital output test (dual-pole output 'SBC+/SBC-': Transistor SBC+ does not switch on) failed
434	Safety card error - Digital output test (dual-pole output 'SBC+/SBC-': Transistor SBC- does not switch on) failed
435	Safety card error - Digital output test (dual-pole output 'SBC+/SBC-': Brake in load, transistor SBC+ does not switch off) failed
436	Safety card error - Digital output test (dual-pole output 'SBC+/SBC-': Brake in load, transistor SBC- does not switch off) failed
437	Safety card error - Digital output test (safe pulse disabler: monitoring error) failed
438	Safety card error - Digital output test (safe pulse disabler: interruption of output stage 'STO' / 'STO SIL3' on start-up) failed
439	Safety card error - Digital output test (output stage branch ('STO') does not switch off) failed
440	Safety card error - Digital output test (output stage branch ('STO SIL3') does not switch off) failed
441	Safety card error - Digital output test (safe pulse disabler: interruption of output stage ('STO' / 'STO SIL3' with cyclical UN) failed
442	Safety card error - ERR02: Motor is running
443	Safety card error – Channel A-B data alignment (monitoring of standstill) failed

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

444	Safety card error – CRC calculation error during the command CS 0: Invalid check sum
445	Safety card error - Safe Stop 2 (SS2): Monitoring of braking ramp failed
446	Safety card error - Digital output test (safe pulse disabler: Interruption of output stage in run mode
447	Safety card error – Voltage monitoring test (monitoring of test pulse current signal) failed
448	Safety card error – Feasibility of position / encoder signal (feasibility with external encoder) failed
449	Safety card error – Feasibility of position / encoder signal (feasibility with setpoint value velocity, different signs) failed
450	Safety card error – Channel A/B data alignment (state of input 'SS1 SIL3/Enable') failed
451	Safety card error – Channel A-B data alignment (monitoring of direction of rotation) failed
452	Safety card error – Channel A-B data alignment (total velocity above 10 ms) failed
453	Safety card error – Channel A-B data alignment (total velocity above 100 ms) failed
454	Safety card error – Channel A-B data alignment (history of current feasibility) failed
455	Safety card error – Channel A-B data alignment (counter fault tolerance current feasibility) failed
456	Safety card error – Channel A-B data alignment (tolerance position error external encoder) failed
457	Safety card error – CRC calculation error during the command CS 0: Invalid string length received
458	Safety card error – Feasibility of position / encoder signal (feasibility with setpoint value velocity, lag distance too great) failed
459	Controller error: Safety relay (AS Enable) was connected by the safety card
460	Safety card error - Safe Brake Test (SBT): Brake test aborted
461	Safety card error - Safe Stop 1 (SS1): Monitoring of braking ramp
462	Safety card error - Timeout: Download of SD memory card
463	Safety card error – Digital input test (SS1 SIL3/Reset) failed
464	Safety card error - Safe Speed Range (SSR): Limit value tolerance range exceeded

4.2 Error messages

465	Safety card error - Safely Limited Speed (SLS): Period length tolerance range exceeded
466	Safety card error - Safely Limited Speed (SLS): Limit value tolerance range exceeded
467	Safety card error - Safe Speed Range (SSR): Period length tolerance range exceeded
468	Controller warning: The NSTOP limit switch has been activated and the motor switched off!
469	Controller warning: The PSTOP limit switch has been activated and the motor switched off!
470	Telnet was closed by [%s] for new connection
471	Multiple CAN devices exist with node number %ld
472	Caution! Telnet [%s] has been closed due to this connection
473	The device with node number %ld is on another CAN network - Command <%c%c>
474	Axis is real and in position control, so cannot execute command %c<%c%c>
475	Invalid value for gantry slave axis <%d> command <%c%c>, value must be greater than master axis
476	Cannot execute command %c<%c%c> because the axis is the slave axis for the gantry controller
477	Error when sending a CAN message, error code: %ld
478	Error when receiving a CAN message, error code: %ld
479	The controller with node number %ld has been disabled
480	Could no longer find the device with node number %ld during the bus scan
481	Start of transmission of drive parameters from node no. %ld to exchange unit
483	Only CAN addresses lower than %ld are permitted; please convert the CAN address of the device with address %ld
484	The device with node number %ld cannot be addressed
485	An automatic configuration has been performed on the device with node number %ld
486	Safety card error – Writing to SD card
487	SafetyNet inactive!
488	SafetyNet error: There was an error in the SDO transmission: Time exceeded
489	SafetyNet error: When writing to the MSG queue
490	Error: Servo controller cannot be activated via SafetyNet
491	Error: Servo controller cannot be deactivated via SafetyNet

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

492	Error: Value DT%d cannot be assigned to a variable
493	Error when generating a drive program, node %ld
494	SafetyNet: Internal error ID %08x
495	No SafetyNet device found
496	Error when initialising Safetynet: ID %ld
497	Safety card error: Safe Direction (SDI) – Monitoring clockwise rotation failed
498	Safety card error: Safe Direction (SDI) – Monitoring in both directions is not permitted
499	Invalid setting on command CD26
500	No SD card inserted on device with address %ld
501	Error occurred while saving to SD card
502	Error occurred while reading from SD card
503	Firmware version of device with address %i is too old
504	SD card on device with address %ld is write-protected
505	Heartbeat error on device with address %ld
506	Could not activate the rotary encoder at address %i
507	The device at address %i is not a rotary encoder
508	Node no. %ld: Too many rotary encoder modules used; maximum of %ld supported
509	Connection with Profibus was disconnected
510	Check sum calculation error when writing to SD card
511	Check sum calculation error when reading from SD card
512	SD card error - Device is too old
513	Safety card error: Safely Limited Increment (SLI) – Standstill threshold exceeded
514	Safety card error: Safely Limited Increment (SLI) – Monitoring of safe increment
515	Safety card error: Safely Limited Increment (SLI) – Monitoring of safely limited speed
516	Safety card error: Safely Limited Increment (SLI) – Monitoring of standstill position
517	Safety card error: Safely Limited Position (SLP) – Monitoring of safe absolute position
518	Safety card error: Safely Limited Position (SLP) – Monitoring of safely limited speed
519	Safety card error: Timing error – Current transformer interrupt is already running
520	Safety card error: Timing error – Cycle time exceeded

4.2 Error messages

521	Safety card error: Feasibility of position / encoder signal - Position jump of external encoder
522	Safety card error: Feasibility of position / encoder signal - Feasibility of absolute position of external encoder
523	Safety card error: Channel A/B data alignment – Current offset
524	Safety card error: Channel A/B data alignment – State of internal feedback signals from safety functions
525	Safety card error: Channel A/B data alignment – State of internal activation signals from safety functions
526	Safety card error: Channel A/B data alignment – Subsequent error
527	<Reserved>
528	Safety card error: Synchronisation channel A/B - Synchronisation current offset
529	<Reserved>
530	Safety card error: Checking of data/parameters - Unknown device type in configuration data
531	Safety card error: Checking of data/parameters – Invalid device version in configuration data
532	Safety card error: CRC error - CRC alignment via configuration data
533	Safety card error: System software error (fatal) – cyclical test expiry monitoring
534	Safety card error: System software error (fatal) – current feasibility expiry monitoring
535	Safety card error: System software error (fatal) – cyclical test execution monitoring
536	Safety card error: System software error (fatal) – input test expiry monitoring
537	Safety card error: System software error (fatal) – CRC test configuration data
538	Safety card error: System software error (fatal) – CRC test of RAM data
539	Safety card error: System software error (fatal) – Safe pulse disabler in non-permitted state
540	Safety card error: System software error (fatal) – Servo amplifier registers non-permitted state of safe pulse disabler
541	Safety card error: System software error (fatal) – input parameter check
542	Safety card error: Test of digital outputs - 'SBC+/SBC-' internal error

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.2 Error messages

543	Safety card error: Test of digital outputs - 'SBC+/SBC-' internal error
544	Error when creating the inputs/outputs of the device with address
545	File does not exist
546	Safety card error: Feasibility of parameters - Holding brake
547	Safety card error: Channel A/B data alignment – 'Ready for operation' state failed
548	Controller warning: Reaction monitoring, controller is disabled
549	ESC - User aborts
550	Command will not be executed as the beta version has expired
551	Position cannot be approached through ST because no positioning command is active
552	ModbusClient: Exception function code received – unknown exception
553	ModbusClient: Connection interrupted; Server is not responding
554	ModbusClient: Error – Connection is being re-established
555	ModbusClient: Connection to Server established
556	ModbusClient: Exception function code received – Illegal function
557	ModbusClient: Exception function code received – Illegal data address
558	ModbusClient: Exception function code received – Illegal data value
559	ModbusClient: Exception function code received – Illegal response length
560	ModbusClient: Exception function code received - Acknowledge
561	ModbusClient: Exception function code received - Slave Device Busy
562	ModbusClient: Exception function code received – Negative Acknowledge
563	ModbusClient: Exception function code received - Memory Parity Error
564	ModbusClient: Exception function code received - Gateway Path Unavailable
565	ModbusClient: Exception function code received - Gateway target device failed to respond
566	ModbusClient: New configuration – Connection is being re-established
567	ModbusClient: Error – Corrupt data received
568	Error FB 'SetConfigData': Invalid value for ConfigData.CycleTime_us
569	Error FB 'SetConfigData': Invalid value for ConfigData.wNumberOfChannels
570	Error FB 'SetConfigData': Invalid value for ConfigData.wNumberForEncoder
571	Error FB 'SetConfigData': Invalid value for ConfigData.wCanCycleTime_ms

4.2 Error messages

572	Error FB 'SetConfigData': Invalid value for ConfigData.wCanNodeAddr
573	Error FB 'SetConfigData': Invalid value for ConfigData.wCan1Baudrate_kBit
574	Error FB 'SetConfigData': Invalid value for ConfigData.wCan2Baudrate_kBit
575	Error FB 'SetConfigData': Invalid value for ConfigData.wCan2CycleTime_ms
576	Error FB 'SetConfigData': Invalid value for ConfigData.wCanMode
577	Error FB 'SetConfigData': Invalid value for ConfigData.wSdCardAddr
578	Error FB 'SetConfigData': Invalid value for ConfigData.wModbusClient
579	Error FB 'SetConfigData': Invalid value for ConfigData.sModbusClientServerIPAddr
580	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientCycleTime_ms
581	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientPrimoReadOffset
582	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientServerReadOffset
583	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientNumberOfReadParameters
584	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientPrimoWriteOffset
585	Error FB 'SetConfigData': Invalid value for ConfigData.diModbusClientNumberOfWriteParameters
586	Error FS Mode asynchronous motor: The drive parameter STOPMODE must have the value 0 (SAVE & COLDSTART is required)

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

4.3 Coded status messages

4.3 Coded status messages

Overview of all the codes that PMCprimo enters in a motor status variable, depending on the axis status.

0 Position control loop closed (PC)
8 Position control loop open, motor switched off (MO)
66 Achse waiting for coupling at the start of a position mapping (XM)
67 Axis executes the coupling process at the start of a position mapping (C).
68 Axis executes an alignment movement for starting a position mapping (A).
128 The interpolator is active for the CNC application (N).
256 Execution of infinite positioning or velocity control (VC)
512 Axis executes a positioning command (MA,MR)
2048 Position mapping (= Map) active (XM)
4096 Axis executes a stop command (ST)
8192 Initialisation is running (IN,IB)

5 Interface: PMCprimo <-> Machine

5.1 Installation guidelines

It is essential that the PMCprimo is installed properly. Protect the PMCprimo from overvoltage and electrical interference signals in the power supply and signal cables. Pilz accepts no responsibility for defects arising from a faulty installation. Please refer to the information in the respective installation manual when installing the PMCprimo devices.

Each PMCprimo axis depends on the signals of the respective rotary encoder. Incorrect rotary encoder pulses produce errors in the absolute position. Great care must be taken when selecting and installing the rotary encoder so that interference can be avoided when the rotary encoder signals are transferred to PMCprimo. The rotary encoder inputs on PMCprimo are isolated by optocouplers. Rotary encoders with potentially isolated line driver outputs can be used. These ensure that signals are transferred safely, even in a rugged industrial environment. Recurring reference signals can be used to avoid an accumulation of errors when transferring rotary encoder signals. Alternatively, rotary encoders can be used, on which absolute positional information is transmitted via an SSI interface.

The digital inputs and outputs on PMCprimo are potentially isolated. Potential isolation guarantees the highest possible immunity to interference signals. Industrial monitoring devices such as a programmable logic controller, for example, can be connected directly without any problem.

The corresponding operating manual should be read when installing.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

5.2 Monitoring functions

To use PMCprimo properly in a machine, the user should consider the following monitoring functions separately for each axis:

- Setting the position error limit with the SE command (see Page 3-106)
- Time monitoring on the rotary encoder signals with the TO command
- Setting software end limits with the LH and LL command
- Defining the software limit switch with the DL command and installing the relevant switches on each axis so that if a limit switch is passed, there is still time to stop the axis before reaching a mechanical limit.

5.3 Encoders

Incremental encoders supply 90° out of phase pulses. These pulses are transmitted on 2 separate tracks. PMCprimo recognises the count direction based on the pulse sequence. In its default setting, PMCprimo expects track B to follow track A for movements in a positive direction. The encoder pulses are evaluated fourfold internally. Rotary encoders with a voltage level of 5V are used as standard. Other voltage levels may be used as an option. Only rotary encoders with potentially isolated line driver outputs should be used. SSI rotary encoders can be used as an alternative to incremental encoders.

It is also possible to use a CANopen rotary encoder. Any bus addresses from 1 to 49 may be set.

5.4 Setpoint output

The setpoint is specified digitally. An analogue CAN output can be used for an analogue setpoint specification.

5.5 Analogue auxiliary output

Each analogue output can supply an analogue voltage level as an analogue auxiliary output. The function is selected using the “**AO**” command. The signal is determined through the commands “**SF**”, “**KM**” and “**OM**”. If activation of the analogue auxiliary output is ended, the voltage level is no longer defined. PMCprimo does not output any voltage in its default setting.

5.6 Ethernet interface

5.6 Ethernet interface

The factory set IP address for the Ethernet interface is: 192.168.0.11.

The outputs from the boot process are displayed after the control system is connected for the first time. The data sent to the control system is returned to the sender as an echo.

5.7 Inputs and outputs

As standard, PMCprimo is equipped with a number of digital inputs and outputs, depending on the device. Inputs and outputs are divided into groups of 8 (Bytes). Inputs and outputs can be programmed for the widest range of purposes. All digital inputs and outputs are potentially isolated and are 24 VDC.

PMCprimo C	8 input groups The first input group is real; only 6 of these inputs can be used. All 6 inputs are fast inputs, which can be used by the commands “**DR**” and “**PS**”.
	8 output groups The first output group is real; only 6 of these outputs can be used.

It is possible to connect a CAN-IO module (e.g. Pilz PSSU) to the PMCprimo CANopen Bus.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

5.7 Inputs and outputs

The two CAN modules “PSSu H CAN” and “PSSu H SB CAN” can also be connected to the PMCprimo CANopen Bus. The following modules are supported:

- PSSu E F PS (but not in R mode)
- PSSu E F PS-1 (but not in R mode)
- PSSu E F PS-P
- PSSu E S 4DI
- PSSu E S 4DO 0.5
- PSSu E S 2DO 2
- PSSu E F 4DI
- PSSu E F 4DO 0.5 (but not in & mode)
- PSSu E F 2DO 2 (but not in & mode)
- PSSu E F 2DOR 8 (but not in & mode)
- PSSu E S 4AI U (but not in X,S mode)
- PSSu E S 2AI U (but not in X,S mode)
- PSSu E S 2AI I (but not in X,S mode)
- PSSu E S 4AO U
- PSSu E S 2AO U
- PSSu E S 2AO I
- PSSu 16DO
- PSSu 15DI
- PSSu 8DO8DI
- PSSu 2DOR2
- PSSu 2DOR10

The inputs/outputs can have any address between 1 and 49 and are addressed as normal inputs/outputs.

The PNOZmulti can also be connected to the PMCprimo CANopen Bus. The inputs/outputs are addressed in the same way as on a PSSU.

PMCprimo user		Comment
`0.1>`	`SO21.1:1`	CAN-IO 1.1 is set
`0.1>`	`DI21.1:2+,PRG`	The input function 'PRG' is defined for CAN-IO 1.2

5.7 Inputs and outputs

The following commands can be used for CAN-IOs:

- MG Define input mask group (see Page 3-165)
- BG Define input inhibit group (see Page 3-166)
- DI Define function input (see Page 3-165)
- II If input true do command line (see Page 3-142)
- WI Wait for input line n in group g (see Page 3-100)
- UI Undefine input definition (see Page 3-148)
- MI Mask function input (see Page 3-144)
- EI Enable function input (see Page 3-148)
- LI List input line definitions (see Page 3-180)
- RI Read input lines in group g (see Page 3-141)
- BI Inhibit function input (see Page 3-146)
- SO Set output line n in group g (see Page 3-138)
- LO List ouptut line definitions (see Page 3-181)
- RO Read output line states in group g (see Page 3-141)
- CO Clear output line n in group g (see Page 3-140)
- IO If output true do command line (see Page 3-143)

Analogue CAN-I/O modules are supported. The modules are assigned to axes automatically. As a result, all existing commands can be used, such as “**DA**”, “**AO**”, “**OM**”, “**KM**” etc., as on existing internal analogue inputs/outputs. Position control is also possible using “**PC**”, “**KP**”, etc.
The analogue inputs/outputs are read or written via synchronous messages. The cycle time is set as previously, using the “**CD**” command.

Up to 16 analogue inputs and 16 analogue outputs can be used. These can be divided over several coupling modules (e.g. 4 analogue inputs on the first module and 8 inputs on the second module).

Due to the connection to the axes, the maximum number depends on the number of axes on the system. It can be set using the “**CD**” command.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

5.8 Drive commands

With PMCprimo it is possible to enter commands for the servo amplifier directly from PMCprimo's PTerm. Drive commands must be set in inverted commas and a connection to the servo amplifier must be established via the "QA" command.

The compiler can store 10,000 drive commands.

Example:

PMCprimo user		Comment
0.1>	"DIR"	Drive command DIR is triggered.
0.1>	DIR	Output from servo amplifier
1		

A drive command may be assigned to a variable. If the servo amplifier does not return a value, an error message is issued.

Example:

PMCprimo user		Comment
0.1:	$VAR="DIR"<CR>	Drive command DIR is assigned to variable.
0.1.	$VAR<CR>	Output from $VAR
0.1:	$VAR=1 (0x00000001)	

When saving with the "**SP**" command, all drive parameters are stored as a program, which is named as follows:

DRIVE_t t: Number of network subscriber

Assigning a value to a variable on a drive command:
e.g.: "GV" = $VAR. The value of $VAR is divided internally by 1000, in order to arrive at floating point numbers. However, it is only divided by 1000 if the command expects a floating point value.

Example: $VAR=2100
"GV"=$VAR produces "GV 2.1"
"PEMAX"=$VAR produces "PE 2100"

Drive command values are assigned to variables as follows:
Through the assignment $VAR = "GV", GV 2.1 supplied the value $VAR = 2100
Through the assignment $VAR = "GV", GV 2 supplied the value $VAR = 2000

5.8 Drive commands

QA t Set ASCII channel for drive

(Set ASCII channel for drive)

Permitted values for t: 0 to 150 (1-50 CANopen, 51-150 SafetyNet)

The address range of 1 to 50 is reserved for CANopen.
The address range of 51 to 150 is reserved for SafetyNet.

The "**QA**" command can be used to access a servo amplifier. The parameter "**t**" determines the address of the required servo amplifier.

On the current axis channel, the drive commands can be used to address the servo amplifier as described above. The command is axis-specific, so it is possible for each axis channel to communicate with a different servo amplifier.

5.8.1 CAN networking

The PMCprimo C supports a maximum of 2 CAN networks, called CAN1 and CAN2. There is just one address range for both networks.

Addresses from 1 to 49 can be assigned.

The baud rates of both CAN buses may be different. (Options 3 and 21 of CD command)

The bus cycle times may also be different. (Options 8 and 23 of CD command)

It is possible to configure one of the two CAN networks as a Slave.

Address 50 is reserved for PMCprimo itself by default.

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

$

$var 3-215
$var> 3-217
$var>$H1...1000 3-218

?

?? 3-197

A

AA 3-90
AB 3-38
Abort 3-38
Abort command execution 3-68
AC 3-205
Acceleration gain 3-115
Acceleration pre-control gain 3-112
AD 3-205
AE 3-179
AH 3-210
AI 3-205
AL 3-210
AM 3-204
Analogue control options word set 3-206
Analogue control proportional gain set 3-205
Analogue input high limit set 3-210
Analogue limit error output define 3-179
Analogue range distance define 3-208
AO 3-159
AP 3-205
AR 3-208
AS 3-69
Assignment variable 3-215
AT 3-207
AV 3-89
AW 3-206
AX 3-68

B

BA 3-201
BC 3-121
BG 3-167
BI 3-146
BK 3-68
BL 3-50
BN 3-170
BO 3-176
Bound overflow count wait for 3-104
Bound position wait for 3-103
BR 3-89
Break out of sequence 3-68
BT 3-202
Bus variable 3-221
BV 3-219

C

CA 3-160
Calculate initial ratio from analogue range distances 3-208
CAN networking 5-260
CAN trace 3-225
CAN-IO-Modul 5-256
CD 3-17
CE 3-70
CF 3-43
CH 3-69
Change channel 3-69
Change to parallel mode 3-70
CI 3-98
CL 3-97
Clear drive fault 3-43
CN 3-225
CO 3-140
command 3-45
Command execution 2-15
Command execution 2-15
Command line 2-15
Command line 2-15
Command overview 2-11
Continuous display mode 3-190
Continuous reference mode on/off set 3-122
Control parameters 3-112
Control word set 3-54
Copyright 1-6
CP 3-70
CR 3-208
CS 3-228
CT 3-96

CW....3-54

D

DA....3-160
DB....3-176
DC....3-46
DD....3-187
DE....3-172
Debounce time reference signal....3-134
Define analogue input type....3-207
Define analogue limit error output....3-179
Define analogue range distance....3-208
Define bound overflow output....3-176
Define error output....3-172
Define function input....3-165
Define hysteresis for position trigger output....3-175
Define hysteresis for velocity trigger output....3-176
Define input inhibit line....3-170
Define input mask group....3-166
Define limit switch input....3-162
Define masked input line....3-168
Define output alignment move....3-177
Define outside window output....3-177
Define position snapshot input....3-163
Define position trigger output....3-173
Define reference accepted output....3-133
Define reference input....3-136
Define reference output....3-134
Define reference reject output....3-178
Define timer/counter clock input....3-153
Define timer/counter gate input....3-154
Define timer/counter output....3-149
Define trigger variable....3-217
Define variable array....3-218
Define velocity trigger output....3-175
Define zero marker input on/off....3-128
DF....3-188
DG....3-188
DI....3-165
Display active programs....3-31
Display actual position....3-187
Display analogue input....3-160
Display calculated reference error....3-188
Display demand position....3-187
Display following error....3-187
Display free memory....3-29
Display last error....3-197
Display last value of trace mode....3-196
Display length reference signal false....3-132
Display length reference signal true....3-133
Display mapped axis bound position....3-189
Display mode off....3-189
Display snapshot position data....3-188
Display time....3-189
Display velocity....3-187
DL....3-162
DM....3-190
DO....3-189
DP....3-187
DR....3-136
Drive commands....5-259
DS....3-188
DT....3-189
DV....3-187
DZ....3-128

E

EI....3-148
EL....3-67
Else - execute commands if expression is false....3-67
EM....3-80
Emergency stop....3-38
EN....3-148
Enable function input....3-148
Enable trigger variables....3-220
Encoder filter....3-53
End of enter map....3-80
End of parallel mode....3-70
End of sequence....3-61
End repeat....3-65
End wait state....3-105
Enter map....3-80
Enter position control mode....3-32
Enter sequence....3-60
ER....3-65
Error options word set....3-110
ES....3-60
EV....3-220
EW....3-110
Execute commands if expression is true....3-66
Execute commands if program is active....3-67
Execute inverse map virtual....3-84

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Execute map 3-82
Execute map virtual 3-83
Execute sequence 3-63
Execute sequence parallel 3-64

F

FC... 3-186
FE..... 3-187
FH.... 3-131
FI..... 3-155
Filter on reference error set 3-125
FL.... 3-131
FM 3-29
FO... 3-155
FR.... 3-125
Free analogue input 3-155
Free analogue output 3-155
Free memory display 3-29
FS... 3-182
FW... 3-129

G

GA... 3-70
GE... 3-71
General information about instruction set 2-11
Get wraparound offset value 3-190
GF... 3-71
Global abort 3-70
Global abort execution 3-71
Global motor off 3-71
Global stop 3-70
Global wait off 3-71
GM.... 3-189
GS... 3-70
GW.... 3-190
GX... 3-71

H

Hiperface..... 3-182, 3-186
HW 3-21

I

IA..... 3-218
IB..... 3-42
If input true do command line 3-142
If motor status true do command line..... 3-143
If output true do command line 3-143
IF... 3-66
II.... 3-142
IN... 3-40
Increment select mode set 3-52
Increment selected parameter 3-53
Inhibit 3-146
Inhibit function input..... 3-146
Inhibit trigger variables..... 3-219
Initial ratio from analogue range distances calculate 3-208
Initialisation position and bounds 3-42
Initialise position 3-40
Installation guidelines..... 5-254
IO... 3-143
IP.... 3-53
IS... 3-52
IT... 3-143
IX.... 3-67

J

JB..... 3-135
JF.... 3-134

K

KA..... 3-115
KD..... 3-114
KF.... 3-115
KI.... 3-114
KM..... 3-157
KP... 3-113
KV..... 3-114

L

LA.... 3-28
LB.... 3-221
LC... 3-152
LE.... 3-197
Length of alignment move..... 3-84
LG.... 3-198
LH..... 3-108

LI....3-180
Limit switch input define....3-162
List all parameters....3-28
List bus variable value....3-221
List input line definitions....3-180
List map....3-81
List map variable value....3-221
List output line definitions....3-181
List sequence....3-62
List trigger variables....3-219
List variable value....3-220
LL....3-108
LM....3-81
LO....3-181
Loop....3-61
LOOP....3-61
LP....3-221
LR....3-125
LS....3-62
LV....3-220
LW....3-94

M

MA....3-33
Map link options word set....3-94
Map link slave axis to differential master axis....3-86
Map link slave axis to master axis....3-85
Map options word set....3-90
Mask input function....3-144
Mask trigger variables....3-219
Master scale factor....3-186
Master timeout....3-98
Master-Slave mode....3-73
Maximum reference correction set....3-124
MB....3-87
ME....3-71
MF....3-88
MG....3-166
MI....3-144
ML....3-85
MM....3-208
MN....3-168
MO....3-32
Monitor output function set....3-155
Monitor output gain set....3-157
Monitoring functions....3-106
Motion commands....3-33
Motion commands....3-33
Motor OFF....3-32
Motor status wait for....3-104
Move at constant velocity....3-39
Move to absolute position....3-33
Move units relative to current position....3-36
MR....3-36
MRAM....3-218
MS....3-186
MT....3-98
MV....3-219
MW....3-90

N

NB....3-186
NL....3-86
NM....3-80
NP....3-215
NS....3-61
NT....3-216
Number of bits for SSI encoder set....3-186

O

OA....3-177
OM....3-157
Operating mode....3-17
OR....3-134
OW....3-177

P

PA....3-199
Parameter commands....3-44
Parameter selected increment....3-53
PC....3-32
PD....3-225
PH....3-175
PO....3-173
Position control word set....3-56
Position from drive....3-225
Position mapping....3-73
Position trigger output define....3-173
Position wait loop relative....3-102
PR....3-22

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Print hardware 3-21
Print help display 3-197
Print to terminal 3-22
PS... 3-163
PT.... 3-53

Q

QA... 5-260
QS 3-224

R

RA... 3-133
RC.... 3-127
RD 3-27
Read input lines in group g 3-141
Reference adjustment forwards output 3-134
Reference correction velocity set 3-126
Reference filter options word set 3-129
Reference input true low limit set 3-131
Reference input wait for 3-103
Reference options word set 3-122
Reference position strange channel 3-135
Relative position wait loop 3-102
Reload stored data 3-27
Repeat command line 3-65
Reset 3-27
Reset to default setup 3-27
RF.... 3-125
RH... 3-134
RI.... 3-141
RJ.... 3-128
RK... 3-135
RL 3-121
RM... 3-122
RN 3-128
RO 3-141
RP.... 3-65
RR 3-178
Rreference input true high limit set 3-131
RS.... 3-27
RT.... 3-109
RV.... 3-126
RW 3-122

S

SA 3-45
Save 3-23
Save parameter 3-23
SB.... 3-120
SC 3-47
Scale mapping 3-88
SD card 3-25
SD... 3-227
SE 3-106
Send SDO 3-224
Set acceleration 3-45
Set acceleration feed-forward gain constant 3-115
Set analogue control mode 3-204
Set analogue control options word 3-206
Set analogue control proportional gain 3-205
Set analogue control setpoint 3-205
Set analogue differential control gain 3-205
Set analogue input high limit 3-210
Set ASCII channel for drive 5-260
Set autostart sequence 3-69
Set auxiliary output channel 3-159
Set backlash compensation distance 3-50
Set braking ramp for AB 3-46
Set clutch length 3-97
Set clutch time 3-96
Set clutch window 3-98
Set continuous reference mode on/off 3-122
Set control word 3-54
Set creep distance 3-47
Set deceleration for ST command 3-46
Set deferred reference adjustment position 3-128
Set differential gain constant 3-114
Set error options word 3-110
Set error variable 3-222
Set filter on reference error 3-125
Set increment select mode 3-52
Set input debounce time 3-176
Set integral gain constant 3-114
Set map base advance 3-201
Set map base offset for master map positions 3-87
Set map base/offset/scale factor adjustment acceleration 3-90
Set map base/offset/scale factor adjustment velocity 3-89
Set map link options word 3-94

Set map options word....3-90
Set map scale factor from bounds ratio....3-89
Set maximum position error....3-106
Set maximum reference correction....3-124
Set monitor output function....3-155
Set monitor output gain....3-157
Set monitor output offset....3-157
Set motor off error sequence....3-71
Set motor status variable....3-223
Set number of bits for SSI encoder....3-186
Set output line n in group g....3-138
Set phase advance for reference signal....3-128
Set phase advance scale factor....3-199
Set position control word....3-56
Set position limit high....3-108
Set position limit low....3-108
Set position overflow bound....3-120
Set proportional gain constant....3-113
Set reference correction velocity....3-126
Set reference error adjustment acceleration....3-127
Set reference error limit....3-125
Set reference filter options word....3-129
Set reference input true high limit....3-131
Set reference input true low limit....3-131
Set reference offset....3-125
Set reference options word....3-122
Set reference repeat length....3-121
Set reference timeout....3-109
Set slave map position offset....3-88
Set slow speed....3-48
Set slow velocity mode....3-49
Set software licence key....3-29
Set status variable....3-223
Set system prompts....3-196
Set timeout....3-107
Set timeout for window....3-51
Set velocity....3-44
Set velocity averaging time constant....3-201
Set velocity feedback gain constant....3-114
Set velocity feed-forward gain constant....3-115
Set virtual motor mode....3-32
Set window....3-50
Set/display position overflow counter....3-121
SF....3-155
SK....3-228
SK....3-29
Slow speed set....3-48
SM....3-88
Snapshot position data display....3-188
SO....3-138
Software update....3-25
SP....3-23
SR....3-124
SS....3-48
SSI....3-182
ST....3-37
Status messages....4-229
Status variable....3-223
Stop....3-37
Store variable in MRAM....3-218
SU....3-25
SV....3-44
SW....3-50
SY....3-225
SY....3-196
System prompts set....3-196

T

TC....3-149
TD....3-196
TF....3-195
TG....3-154
TH....3-195
TI....3-51
Time step for trace mode....3-192
TK....3-153
TL....3-195
TM....3-85
TO....3-107
TP....3-196
TR....3-191
Trace CAN....3-225
Transfer map data....3-85
Trapezoidal velocity profile....3-34
TT....3-192
TW....3-192
TZ....3-154

U

UI....3-148
UL....3-86
Undefine input definition....3-148

Pilz GmbH & Co. KG, Felix-Wankel-Str. 2, 73760 Ostfildern, Germany
Telephone +49 711 3409-0, Telefax +49 711 3409-133, E-Mail: pilz.gmbh@pilz.de

Undefine output definition 3-149
Unlink slave axis from master axis 3-86
UO... 3-149
Update 3-25

V

Variable assignment 3-215
Variable number program assignment 3-215
VC... 3-39
Velocity averaging time constant set 3-201
Velocity pre-control gain 3-112
Velocity reverse amplification 3-112
Velocity trigger output define 3-175
VH 3-176
Virtual inputs 3-137
Virtual outputs 3-137
VJ 3-49
VM 3-32
VN 3-20
VO... 3-175
VT... 3-201
VX... 3-219

W

WA 3-101
Wait for absolute position 3-101
Wait for bound overflow count 3-104
Wait for bound position 3-103
Wait for input line n in group 3-100
Wait for motor status 3-104
Wait for reference input 3-103
Wait for time 3-99
Wait for write to variable 3-216
Wait loops 3-99
WB... 3-103
WC... 3-104
WE 3-105
WF 3-103
WI 3-100
WR... 3-102
Write to variable wait for 3-216
WS 3-104
WT 3-99
WV... 3-216

X

XA 3-46
XI 3-84
XM 3-82
XP 3-64
XR 3-209
XS 3-63
XV 3-83
XX 3-84

Z

ZC 3-119
Zero position counters or set position 3-119
ZH 3-131
ZL 3-131
ZT 3-31
ZW 3-56
ZX 3-132
ZY 3-133

© Pilz GmbH & Co. KG, 2008

In some countries, InduraNET p®, Pilz®, PIT®, PMCprotego®, PMI®, PNOZ®, Primo®, PSEN®, PSS®, PVIS®, SafetyBUS p®, SafetyEYE®, SafetyNET p®, the spirit of safety® are registered, protected trademarks of Pilz GmbH & Co. KG. Text and graphics in this publication are simply intended to give an overview of the system. No responsibility accepted for errors or omissions.

60 1948-2008 AUTOMATION

ÖKOPROFIT

Deutscher Akkreditierungs Rat
DAR
DAT - I - 010/05

CERTIFIED DIN EN ISO 9001 DQS QUALITY SYSTEM REG. NO. 3462

...
In many countries we are represented by our subsidiaries and sales partners.

Please refer to our homepage for further details or contact our headquarters.

Pilz GmbH & Co. KG
Felix-Wankel-Straße 2
73760 Ostfildern, Germany
Telephone: +49 711 3409-0
Telefax: +49 711 3409-133
E-Mail: pilz.gmbh@pilz.de

www
www.pilz.com

Technical support
+49 711 3409-444

pilz